Ultra Low Power Multi-gigabit Digital CMOS Modem Technology for Millimeter Wave Wireless Systems

A Thesis
Presented to
The Academic Faculty

by

Ashwin K. Muppalla

In Partial Fulfillment
of Requirements for the Degree
Master of Science in
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2010

Ultra Low Power Multi-gigabit Digital CMOS Modem Technology for Millimeter Wave Wireless Systems

Approved by:

Dr. Joy Laskar, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Manos Tentzeris
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 6th May 2010

I dedicate all my research work and the culmination, this thesis:
to my Mom, Dad, Grandmother
and my brother Anil.

ACKNOWLEDGEMENTS

I would like to thank my graduate advisor Dr. Joy Laskar for his inspiring leadership and guidance throughout the course of my research work. I would also like to express my gratitude to Dr. Saibal Mukhopadhyay and Dr. Manos Tentzeris for taking the time and serving on my reading committee.

I wish to acknowledge Dr Stephane Pinel, Dr. Bevin Perumana and Dr. Padmanava Sen for their technical guidance and support throughout my research work. I wish to thank Dr. Saikat Sarkar for his constant support and valuable friendship.

I take this opportunity to thank all the team members of the Millimeter-Wave Applications Group, and the staff of Georgia Electronic Design Center (GEDC). I thank Chan-Kim and Jin-Keyong Kim from ETRI, Korea, for their guidance during the design of PHY and MAC modules. I would like to thank Gopal Iyer, Kevin Chuang and Patrick Melet for their constant support and friendship. I would also like to thank Jyothsna Krishna Rau for being part of this journey and a source of encouragement.

None of this would be possible without the unwavering support and blessings of my parents, my brother and Lord Ganesha and Lord Narasimha.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

SUMMARY

An ASIC design methodology for sub nanometer nodes is presented. A robust ASIC design methodology is important for any chip design framework. A standard cell library for 45nm and 90nm technology is also presented. Together the ASIC design flow and the standard cell library form an integral part of any chip design framework. Baseband modules such as PHY and MAC with a combined gate count of greater than one million gates are successfully implemented using the ASIC design methodology at 90nm.

A coherent and non-coherent demodulation system as part of the RF module is presented for various modulation schemes. The design and implementation of the demodulation system is presented in great detail. The demodulation system fits into the multi-chip 60 GHz millimeter wave system. The demodulation system is implemented using 90nm process technology and the measurement results in the form of eye diagrams and frequency spectra are also presented for different data rates.

In case of a multi chip 60 GHz millimeter wave solution a SerDes module is required to interface between the RF front end and the Digital Baseband module. The design and implementation of 16 bit and 20 bit SerDes module is also presented. Implementation of the SerDes module using the ASIC design methodology developed in the first part of this work is presented.

Therefore this work encompasses critical modules in the multi-chip CMOS 60 GHz millimeter wave solution i.e. PHY/MAC implementation as part of the Baseband chip and coherent and non coherent demodulation block for the RF chip and the SerDes module as the interface between the baseband and the RF chip.

1 INTRODUCTION

1.1 Motivation

The tremendous impact Silicon based RF technology on wireless technology is unquestionable. It has made possible for consumers to access voice/data and entertainment in virtually every part of the globe, with a variety of accessing platforms i.e. from short range Bluetooth and Wi-Fi networks to satellite and cellular networks, depending on the range and throughput requirements. But there still exists an obvious missing link between the various portable devices such as cell phones, digital cameras, music and video players etc. One potential candidate for the missing link is Nascent UWB (Ultra Wide Band). It however suffers from problems with interference and limited data rate. Moreover UWB systems have very limited transmission power, meaning the bandwidth needs to be traded to overcome the limited SNR (Signal to Noise ratio). Therefore there exists a need for an efficient millimeter wave transmission technology to form the missing link between existing wireless technologies [1].

1.2 Choice of CMOS based 60 GHz millimeter wave technology

The maximum data rate of a communication channel according to Shannon's theorem [2][3] is given by

$$C = BW \log_2(1 + SNR)$$

It is clear from Shannon's theorem that the maximum data rate or channel capacity of a Channel can be increased by increasing the bandwidth. Since information is modulated using a carrier signal, more bandwidth is available if carrier signals of higher frequency are used. Moreover the attenuation is higher at higher frequencies, resulting in the reduction in the level of interference. Therefore this reduction in level of interference coupled with the use of coding and modulation techniques available result in the increase of SNR thereby further increasing the communication data rate. Thus making the use of millimeter wave wireless communication a logical choice.

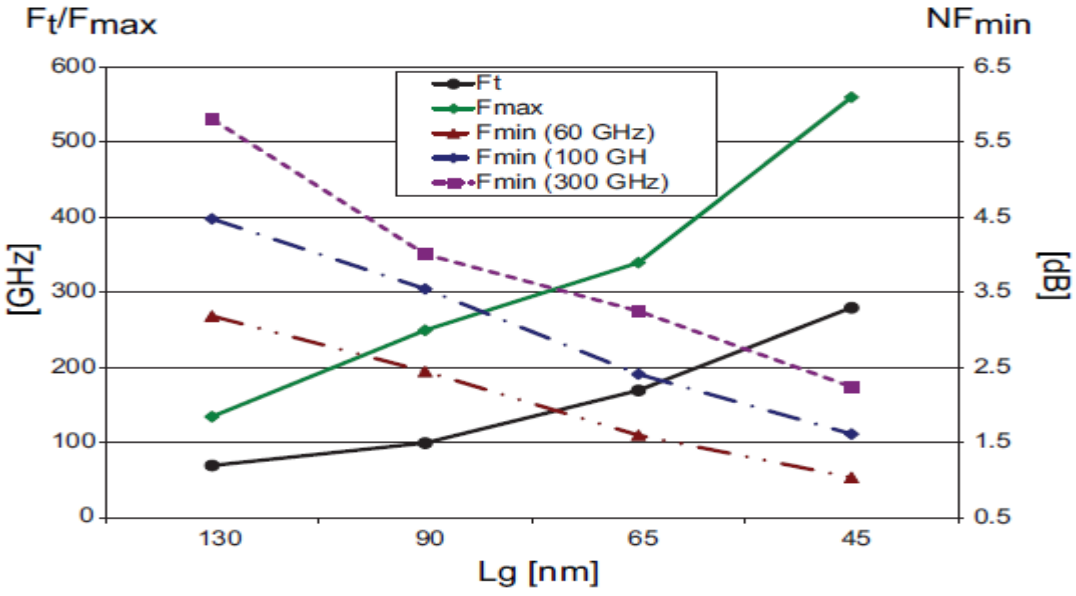


Figure 1: f_T/f_max curve for different CMOS process nodes

CMOS technology has been the platform of choice for low cost high speed digital designs such as microprocessors, consumer electronics etc. Driven by the need for more digital computation and memory CMOS technology has seen aggressive scaling. The f_T/f_{max} curve Figure 1 [5] indicates that a steady performance improvement is expected as a result of scaling of CMOS technology. It can be seen from Figure 1 that the transit frequency f_T at 90nm is $\cong 100Ghz$, at 65nm is $\cong 165Ghz$ and 45 nm is $\cong 280Ghz$. Therefore current CMOS technology nodes can easily support frequency requirement of millimeter wave circuits. In addition to this, CMOS technology has low cost per function when compared to other technologies thereby making it an attractive option for millimeter wave circuits.

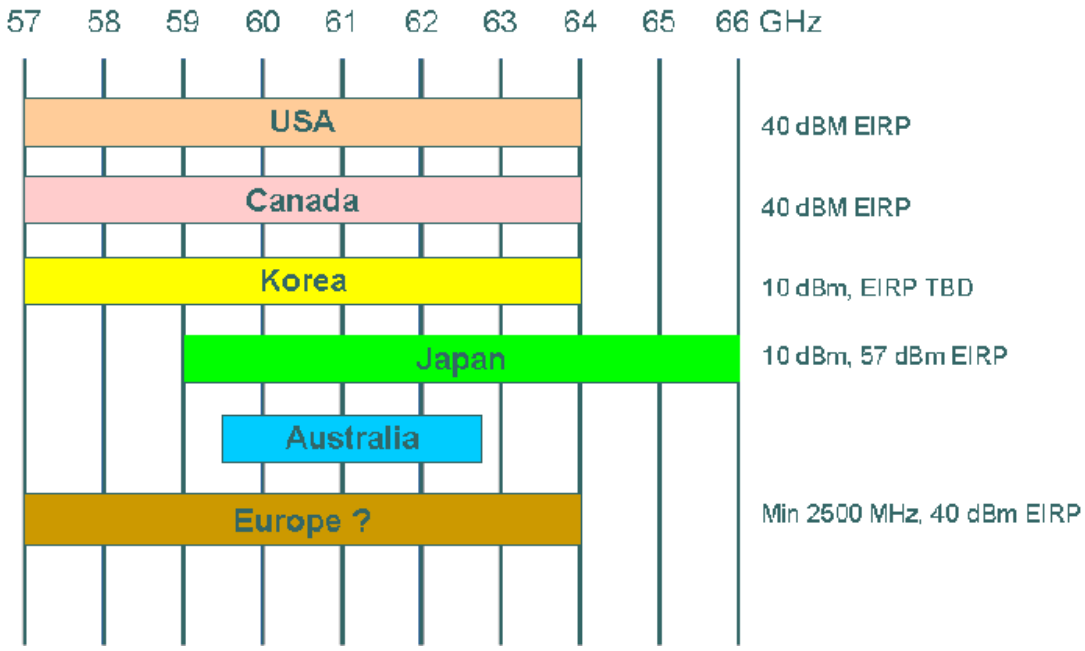


Figure 2: Spectrum allocation

In 2001 the Federal Communications Commission (FCC) [6] allocated a substantial block of 7Ghz in the 57-64Ghz band [4]for unlicensed use. The fact that this bandwidth is unlicensed means that operators are not required to spend significant amount of time and money in obtaining a license. More importantly this bandwidth around 60 GHz is available throughout the world thereby making standardized applications for 60 GHz possible. This availability of unlicensed bandwidth around 60 GHz in addition to the advancements in CMOS technology scaling makes CMOS based 60 GHz millimeter wave circuits an attractive proposition.

2 ASIC DESIGN METHODOLOGY

A robust ASIC design methodology forms an integral part of any chip design framework. Therefore it is essential to have a validated design methodology that can be used to greatly simplify the design process. The various aspects of ASIC design methodology are as follows: [7]

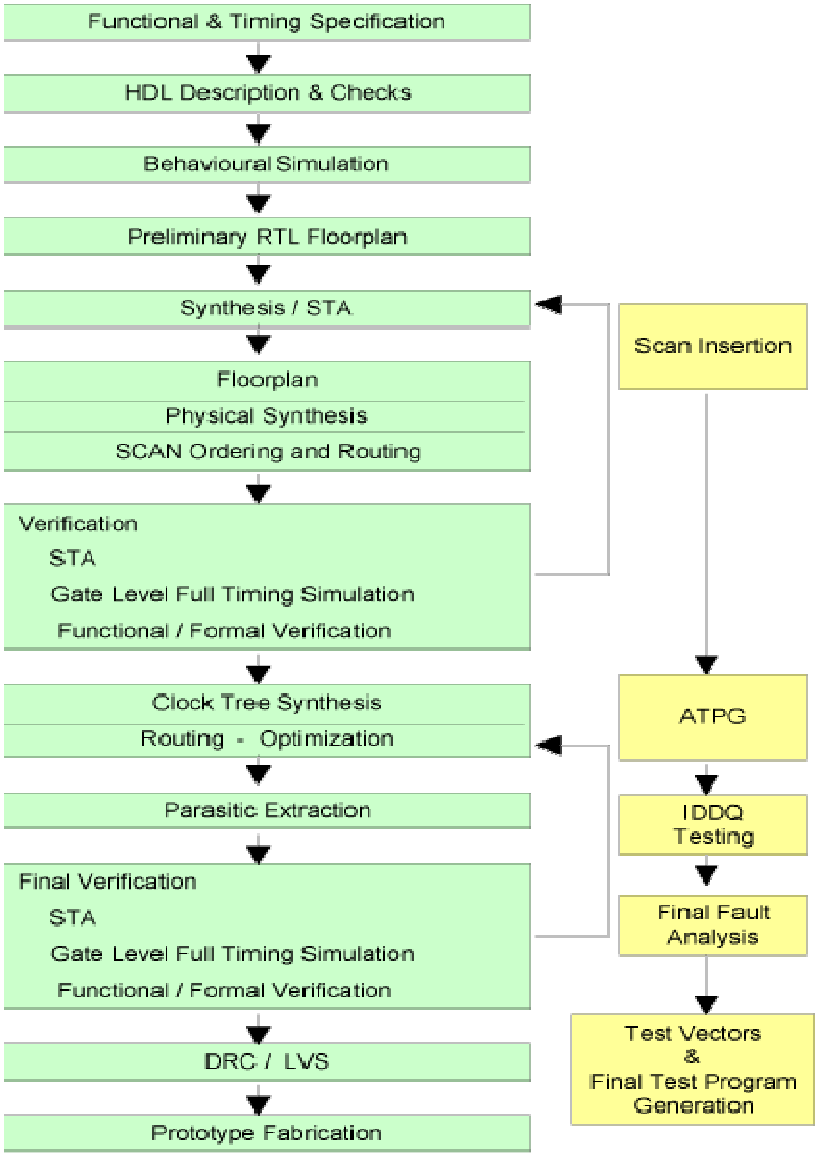


Figure 3: ASIC Design Flow

2.1 Design Specification and HDL coding

Chip design process starts with a high level specification of the design that is to be implemented. This specification typically consists of description of the functionality of the design. It may also split the design into functionally independent blocks i.e. partitioning the design based on functionality into smaller more manageable blocks. Once the design is completely specified it is now converted into RTL (Register Transfer Level) description using HDL (Hardware Descriptive Language) such as Verilog [8], VHDL [9] etc. This RTL is then later used to convert the design into logic gates.

2.2 Pre Silicon Simulation (Functional Simulation)

Once the RTL has been written, it is essential to run a dynamic simulation to ensure that the intended functionality has been accurately translated to the RTL. This dynamic simulation is typically done using tools such as Modelsim [10], NcSim [11] etc. The design under test is excited with inputs from a test bench and the output of the design to these inputs is observed and compared against the expected output to check if the RTL has been written correctly.

2.3 Logical Synthesis

Once the RTL has been verified using the dynamic simulation, the RTL is now converted to a gate level schematic. This is done by using tools such as Design Compiler [12].

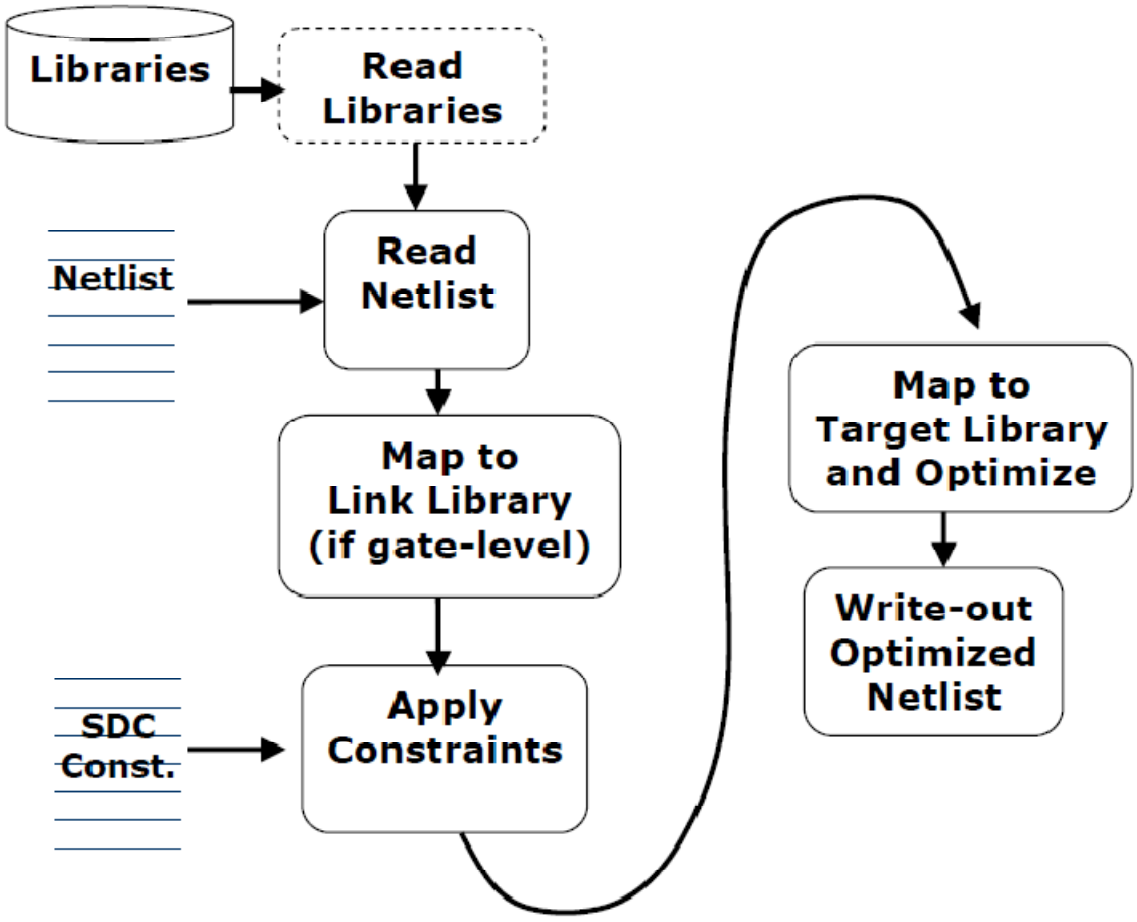


Figure 4: Logical Synthesis Block Diagram

In order to convert the RTL to a functional gate level design it is important to provide the required collateral to the Synthesis tool. The collaterals for synthesis are:

i. RTL source files.

ii. Library Files: These are files that describe the timing, power and functionality of the standard cell library that is used to implement the design. Liberty (lib) [13] format is a popular format for library files.

iii. Constraints: These constraints could be for timing convergence or for meeting certain DRC (design rule check) specifications.

In order to accurately convert the RTL to a gate level design it is essential to appropriately constrain the design during synthesis. There are several types of constraints such as:

i. **Min and Max Library:** In order to ensure that the design meets both setup and hold time requirements, it is important to mention both min and max PVT (Process, Voltage and Temperature) corner libraries so that the Synthesis tool can use min library to estimate and meet hold time requirements and max library to estimate and meet setup time requirements, since min corner is the worst case for hold time and max corner is the worst case for setup time.

Design Compiler (DC) command:[12]

set_min_library Worst_Case_library -min_version Best_Case_library

ii. **Constraints for timing convergence:** These are constraints that are required for ensuring that the implemented design meets the required timing specification in terms of clock frequency and latency.

Clock Information : It is required to define a clock port and the corresponding clock period and information associated with the clock such as clock uncertainty,

clock period etc so that the synthesis tool can use this clock information to appropriately fix setup and hold timing requirement.

DC command: [12]

create_clock [get_ports clock] -period 0.40

set_clock_uncertainty <uncertainty number> [get_clocks clock]

set_dont_touch_network clock

set_clock_latency -source <latency> [get_clocks clock]

IO delay information: It is also required to mention the IO delays while synthesizing the design because this information will indicate to the synthesis tool the amount of time that is available for the logic within the design, so that the tool can optimize accordingly. This information is very essential when working with designs which have hierarchical implementation.

DC command: [12]

set_output_delay 0 -clock [get_clocks clock] <ouput_port_name>

set_input_delay 0 -clock [get_clocks clock] <input_port_name>

Design Rule Constraints: It is also important to specify the required design rule constraints. These rules are generally specified in the technology library and are specific to a particular process technology. The design rules are required to be met in order to guarantee functioning silicon.

DC command [12]

set_max_transition <value> <object list>

set_max_capacitance <value> <object list>

set_max_fanout <value> <object list>

Once the design constraints have been applied the synthesis tool now converts the RTL code to the gate level schematic which satisfies the timing requirements as specified earlier.

DC command [12]

compile_ultra -timing_high_effort_script -no_seq_output_inversion -no_boundary_optimization

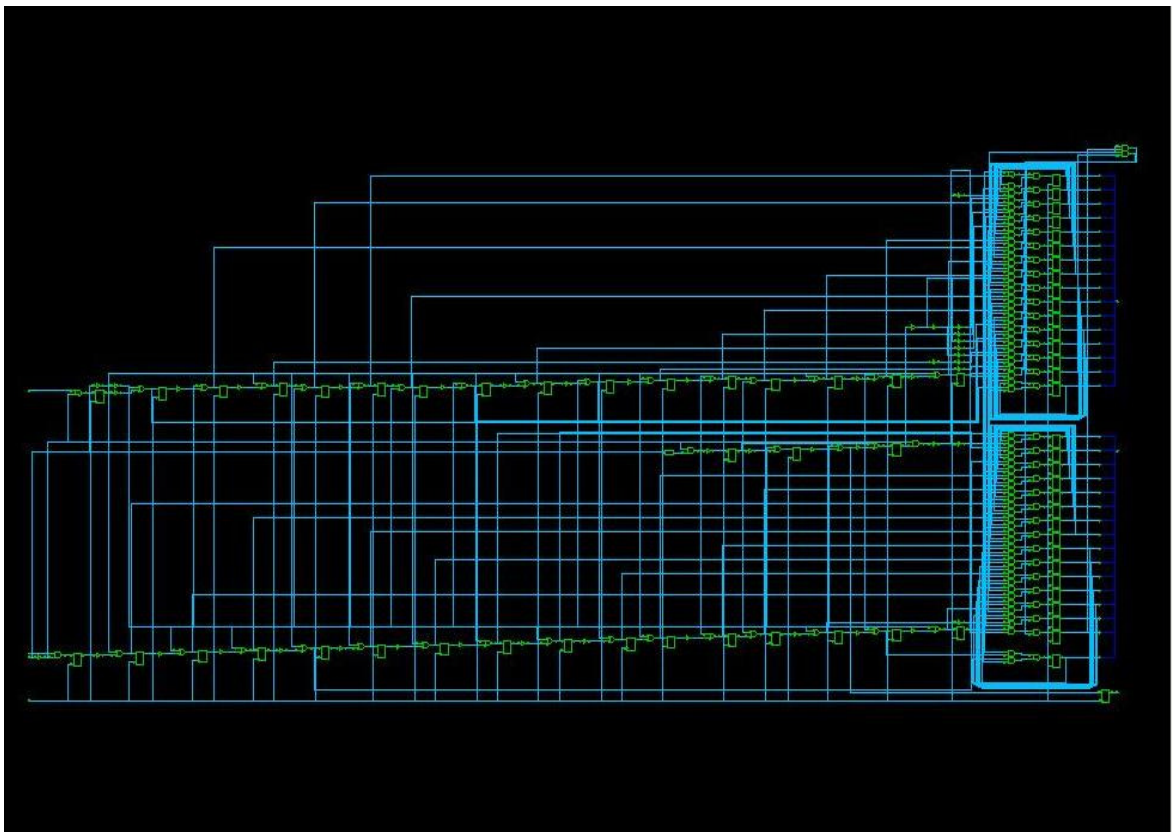


Figure 5: Post Synthesis Schematic

```
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : s2p
Version: B-2008.09-SP3
Date   : Sun Apr 25 08:02:39 2010
****************************************

Operating Conditions: wc_0.90V_125C_10y   Library: CORE90GPLVT
Wire Load Model Mode: enclosed

  Startpoint: ld_reg (rising edge-triggered flip-flop clocked by rx1728clk)
  Endpoint: data_i_reg[5]
            (rising edge-triggered flip-flop clocked by rx1728clk)
  Path Group: rx1728clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  s2p                area_2Kto3K           CORE90GPLVT

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock rx1728clk (rise edge)              0.00       0.00
  clock network delay (ideal)              0.00       0.00
  ld_reg/CP (FD1QLVTX2)                    0.00       0.00 r
  ld_reg/Q (FD1QLVTX2)                     0.11       0.11 r
  U278/Z (ND2LVTX2)                        0.05       0.16 f
  U283/Z (IVLVTX6)                         0.05       0.21 r
  U160/Z (IVLVTX12)                        0.03       0.24 f
  U206/Z (ND2LVTX2)                        0.04       0.28 r
  U203/Z (ND2LVTX4)                        0.03       0.31 f
  data_i_reg[5]/D (FD1QLVTX4)              0.00       0.31 f
  data arrival time                                   0.31

  clock rx1728clk (rise edge)              0.30       0.30
  clock network delay (ideal)              0.00       0.30
  clock uncertainty                       -0.01       0.29
  data_i_reg[5]/CP (FD1QLVTX4)             0.00       0.29 r
  library setup time                      -0.11       0.18
  data required time                                  0.18
  -----------------------------------------------------------
  data required time                                  0.18
  data arrival time                                  -0.31
  -----------------------------------------------------------
  slack (VIOLATED)                                   -0.13


1
dc_shell>
```

Figure 6: Design Compiler Timing Report

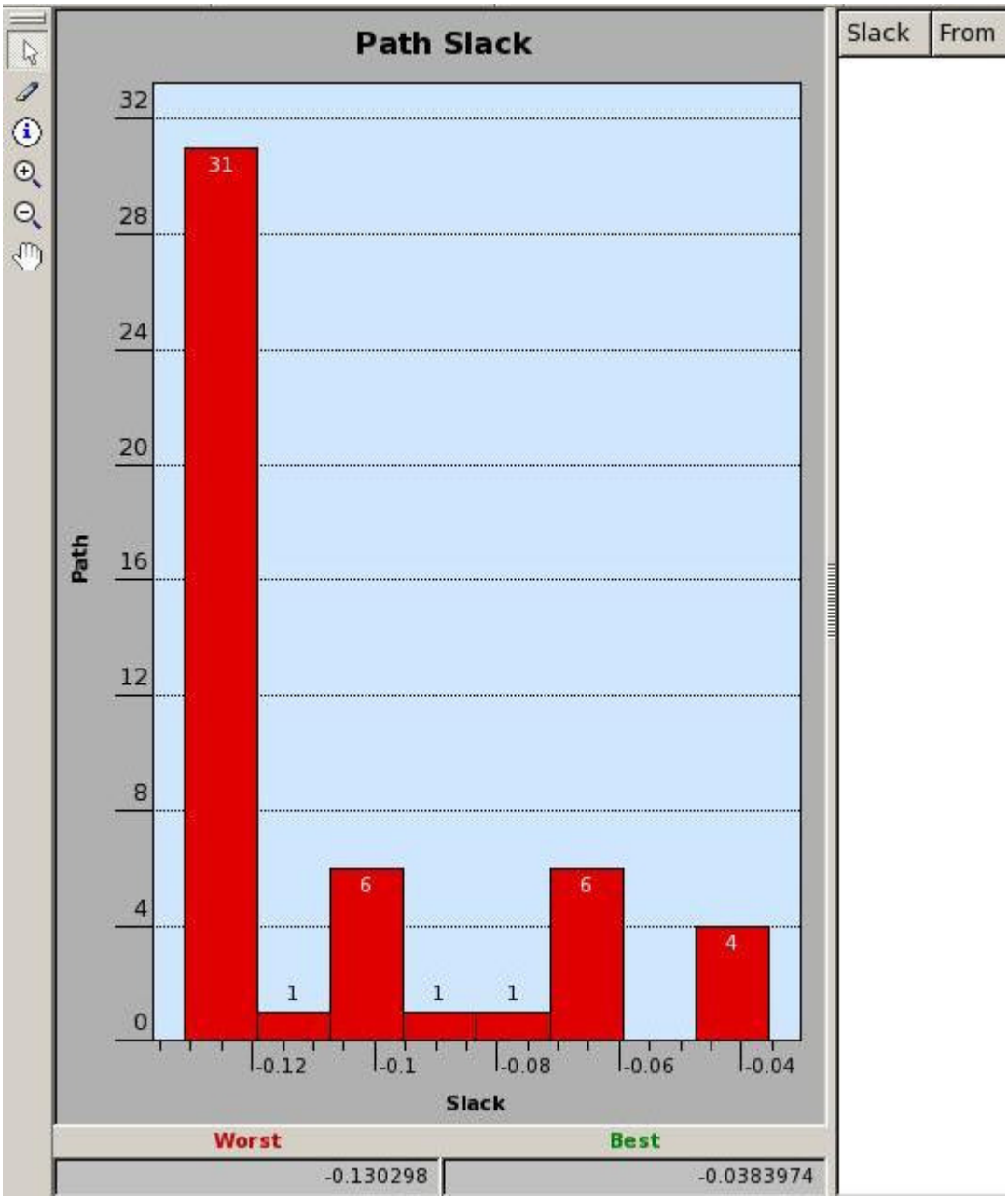


Figure 7: Design Compiler Timing Histogram

2.4 Floor Planning

The main objective of floor planning is to ensure that the design occupies minimum possible area while meeting the specified timing constraints. In floor planning one ensures that the power rails

for VDD and GND are laid down around the chip, and also develop a power grid within the design for better power delivery. During the floor planning stage it is important to ensure that the logical cells and the hard macros such as memory (ROM and RAM) and other IP's (intellectual property) in the form of hard macros are optimally placed as a result of which a minimum net length is required to connect these blocks to the rest of the design. Therefore this ensures that the RC delays of such nets do not dominate the path delays. It is common to have multiple iterations of floor planning and the rest of the layout process before finally achieving optimal placement positions for the design. Another important aspect of floor planning is the placement of IO pins on the design boundary. It is important to ensure that the IO pins are placed correctly in order to achieve quick timing closure.

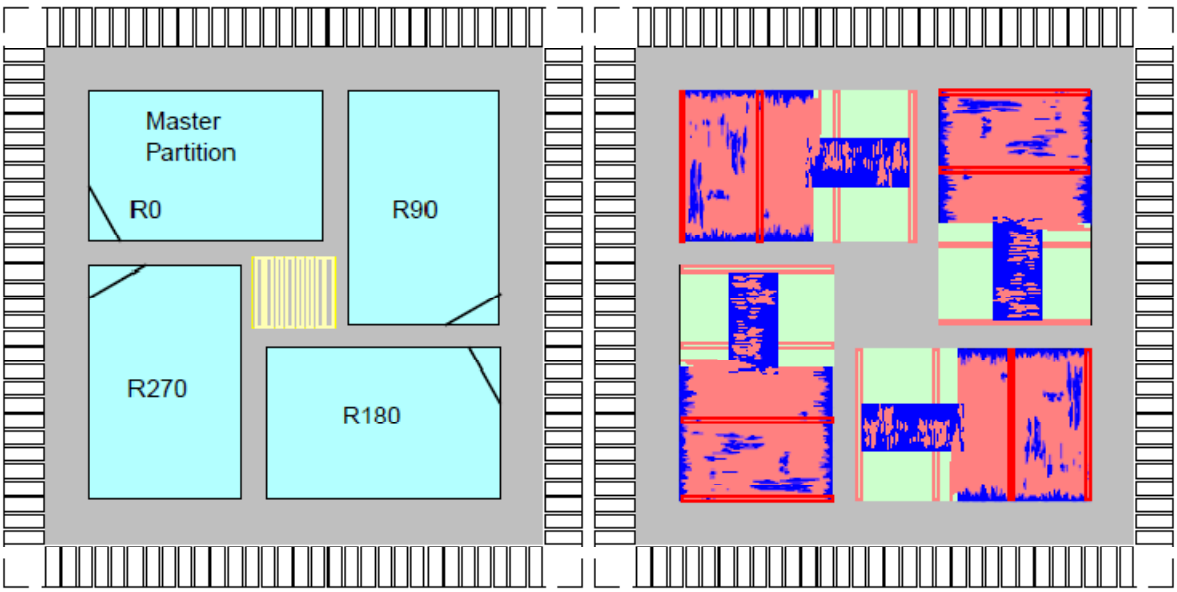


Figure 8: Partitioning a design in Encounter

In case of large designs floor planning helps in breaking down the design into more manageable blocks. The quality of the floor planning process itself has a strong impact on the routing

congestion in the design at the routing phase of ASIC design methodology, therefore it is important to achieve a good floor plan in order to ensure that the design converges both in timing and congestion once it has been implemented. [14]

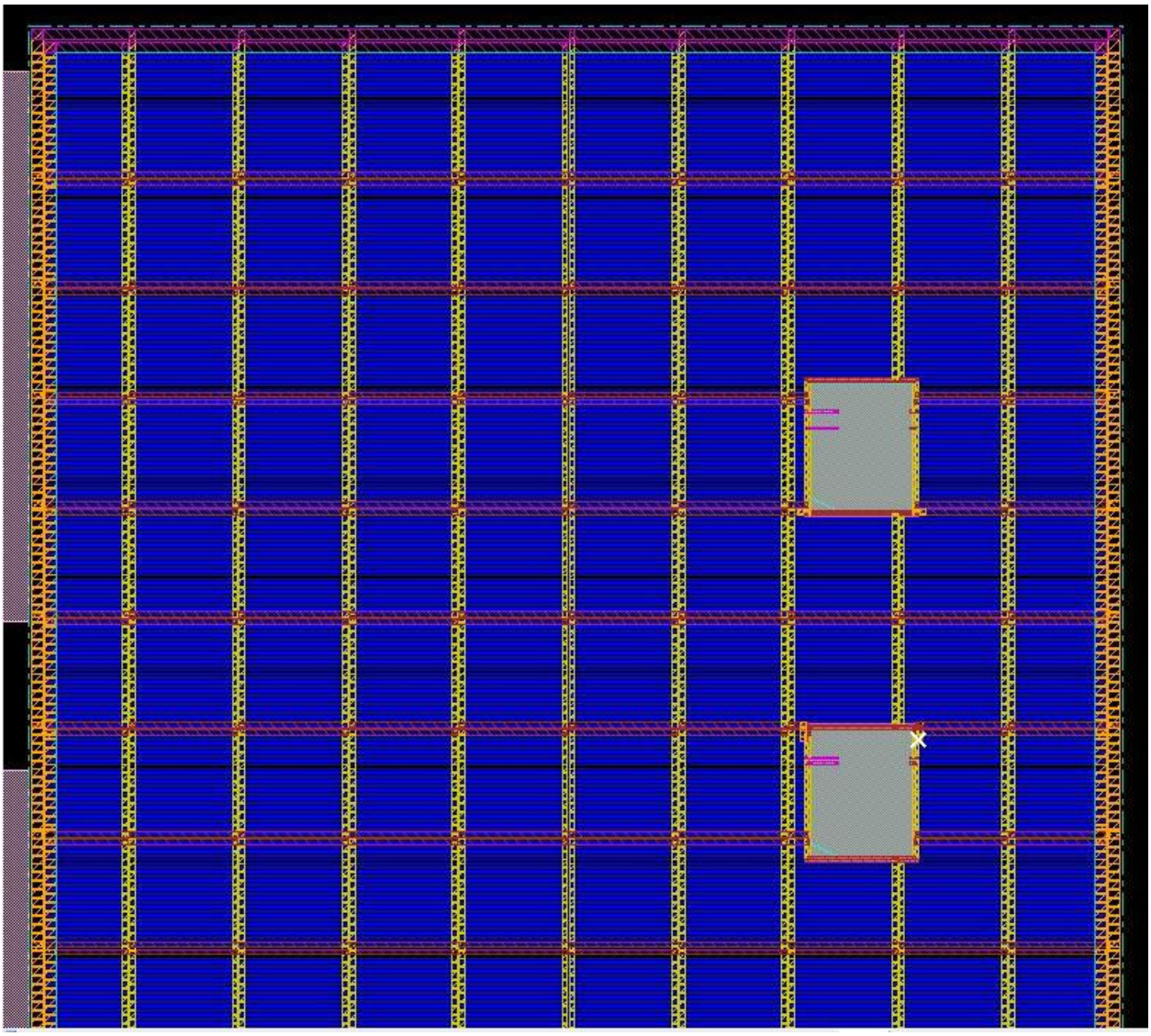


Figure 9: Creating Power Ring and Power Grid for PHY in Encounter

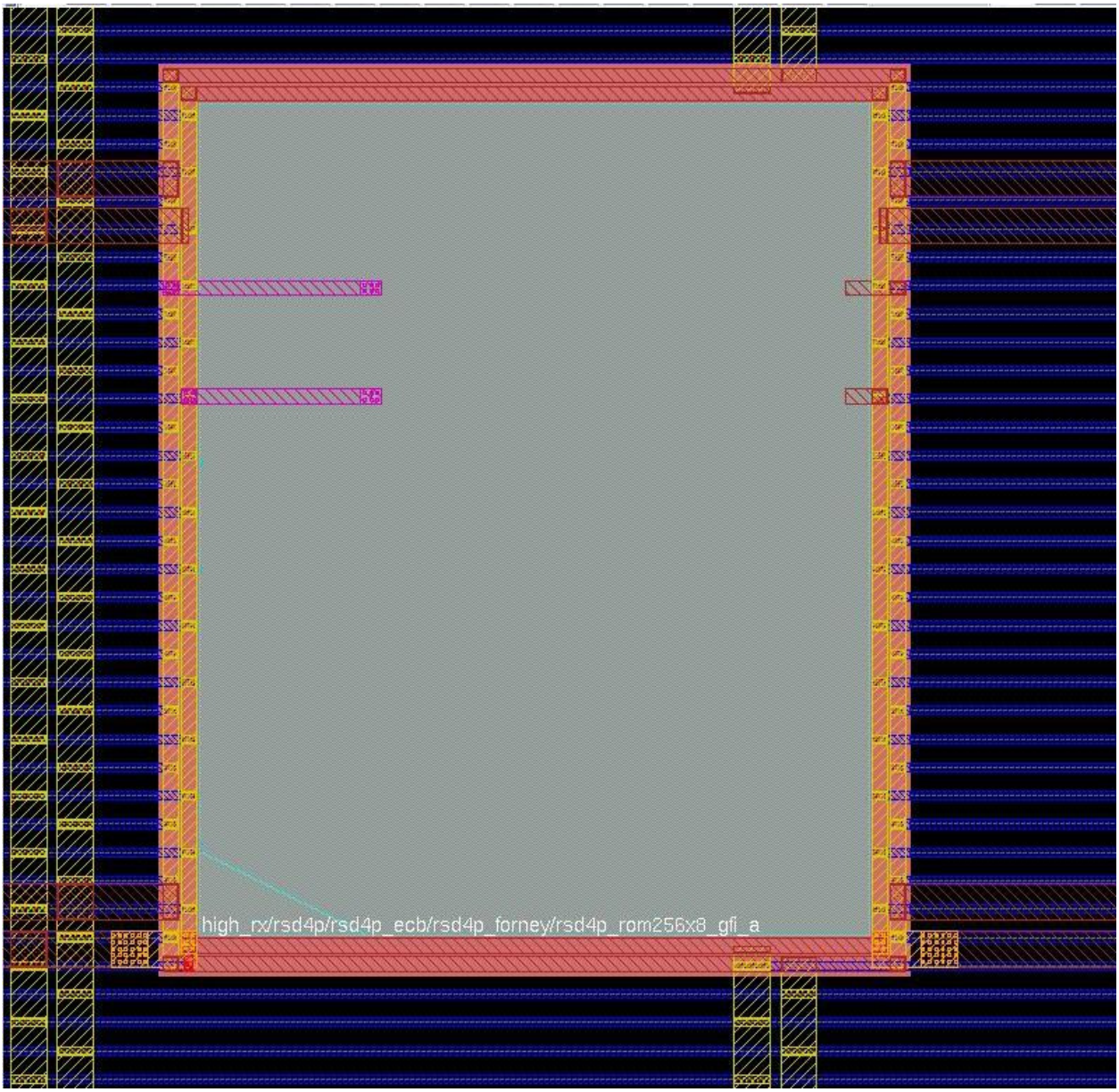


Figure 10: PHY Hard Macro placement in Encounter

2.5 Physical Placement (Timing Driven)

Physical Placement is the process of placing the design cells in appropriate locations based on their connectivity with the rest of the design while ensuring that the timing

requirements are met. In order to ensure that the correct constraints are applied during the placement process, the design constraints from the logical synthesis step needs to be carried forward. This is done by using the sdc [15] (Synopsys design constraint) file. This file contains all the timing, DRC and clock information and is subsequently applied on the design during the placement process once this file has been read.

The inputs for placement tools such as Cadence Encounter [14] are as follows.

i. Synthesized netlist from Logical Synthesis.

ii. Technology libraries for Timing analysis during placement.

iii. Design Constraint file as sdc (Synopsys design constraint) file consisting design constraints such as clock definition, IO delay constraints etc.

For example: Format of sdc file

create_clock [get_ports clock] -period 0.4 -waveform {0 0.2}

set_clock_latency -source 0 [get_clocks clock]

set_clock_uncertainty 0.015 [get_clocks clock]

set_input_delay -clock clock 0 [get_ports {real_dataa[2]}] …

iv. Technology Lef (layout exchange format) [16] files in order to ensure that the placement of the cells does not violate DRC requirements of technology such as spacing, metal width etc.

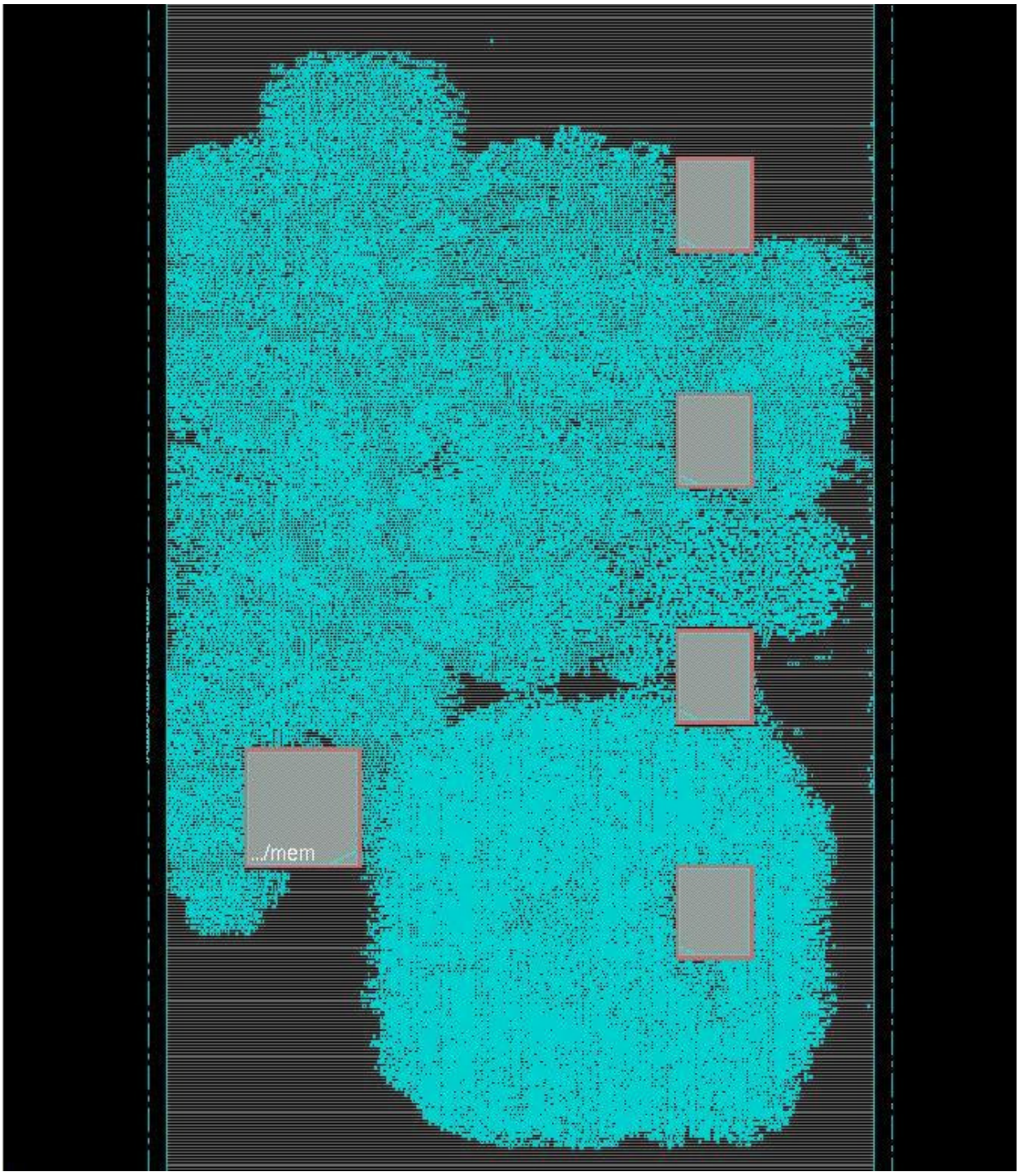


Figure 11: PHY Physical placement in Encounter

2.6 Clock Tree Synthesis

Clock Tree Synthesis is required to ensure that the clock is delivered to the sequential cells (flops / latches) with minimum skew and latency.

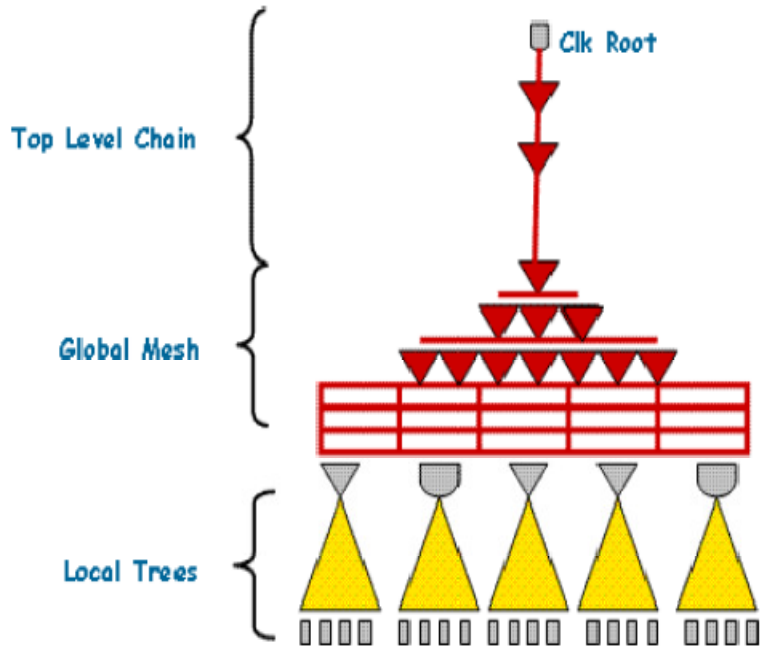


Figure 12: Clock tree Structure

Latency can be defined as the time delay from the clock source to the clock sink. **Skew** can be defined as the difference in the clock arrival times at different clock sinks. It is possible to achieve minimum skew and latency with inputs from the designer. These inputs are typically the type of buffers that need to be used in the clock tree, the target skew and insertion delay of the clock tree. In order to reduce the clock latency high drive strength buffers need to be used in the clock tree.

In case of multi clock designs it is important to minimize the skew across clock domains, since fixing timing paths between asynchronous clock domains is already challenging, and having a bad skew after CTS will not help the cause. Therefore it is important to balance the clock tree for different clock domains i.e. same number of CTS buffers.

The inputs to CTS such as target skew, latency and types of CTS buffers to be used in addition to the inputs from physical placement is exemplified below

CTS input for clock

```
File  Edit  View  Terminal  Tabs  Help
#
#
AutoCTSRootPin  rstclk_gen/i_3/Z
NoGating        rising
Buffer <Buffer Cellname>
MaxDelay        200ps
MinDelay        0ps
MaxSkew         50ps
End
#
#
#
AutoCTSRootPin  rstclk_gen/u0/Z
NoGating        rising
Buffer  <Buffer Cellname>
MaxDelay        200ps
MinDelay        0ps
MaxSkew         50ps
End
#
#
AutoCTSRootPin  rstclk_gen/i_1/Z
NoGating        rising
Buffer  <Buffer Cellname>
MaxDelay        200ps
MinDelay        0ps
MaxSkew         50ps
End
#
#
#
#
#
AutoCTSRootPin  rstclk_gen/u1/Z
NoGating        rising
Buffer  <Buffer Cellname>
MaxDelay        200ps
MinDelay        0ps
MaxSkew         50ps
End
```

Figure 13: Snapshot of Clock tree description for PHY file in Encounter

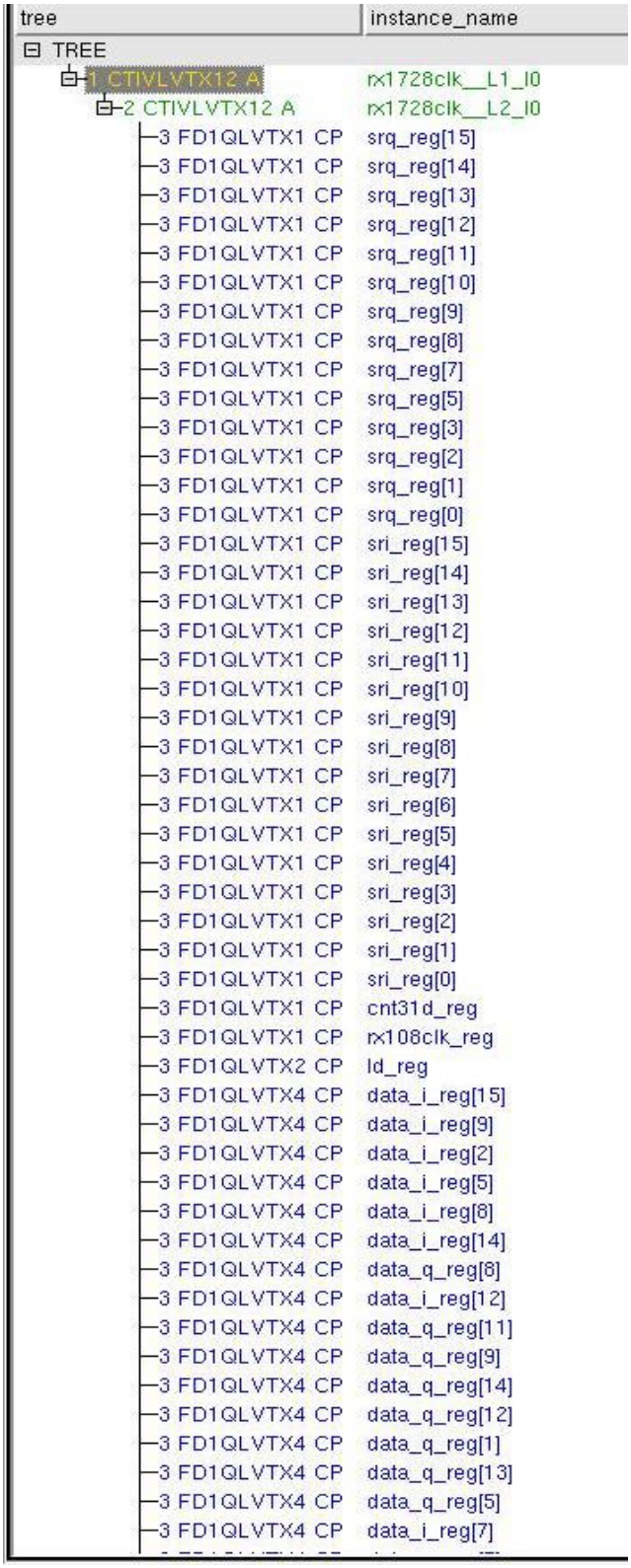


Figure 14: PHY Clock Tree browser in Encounter

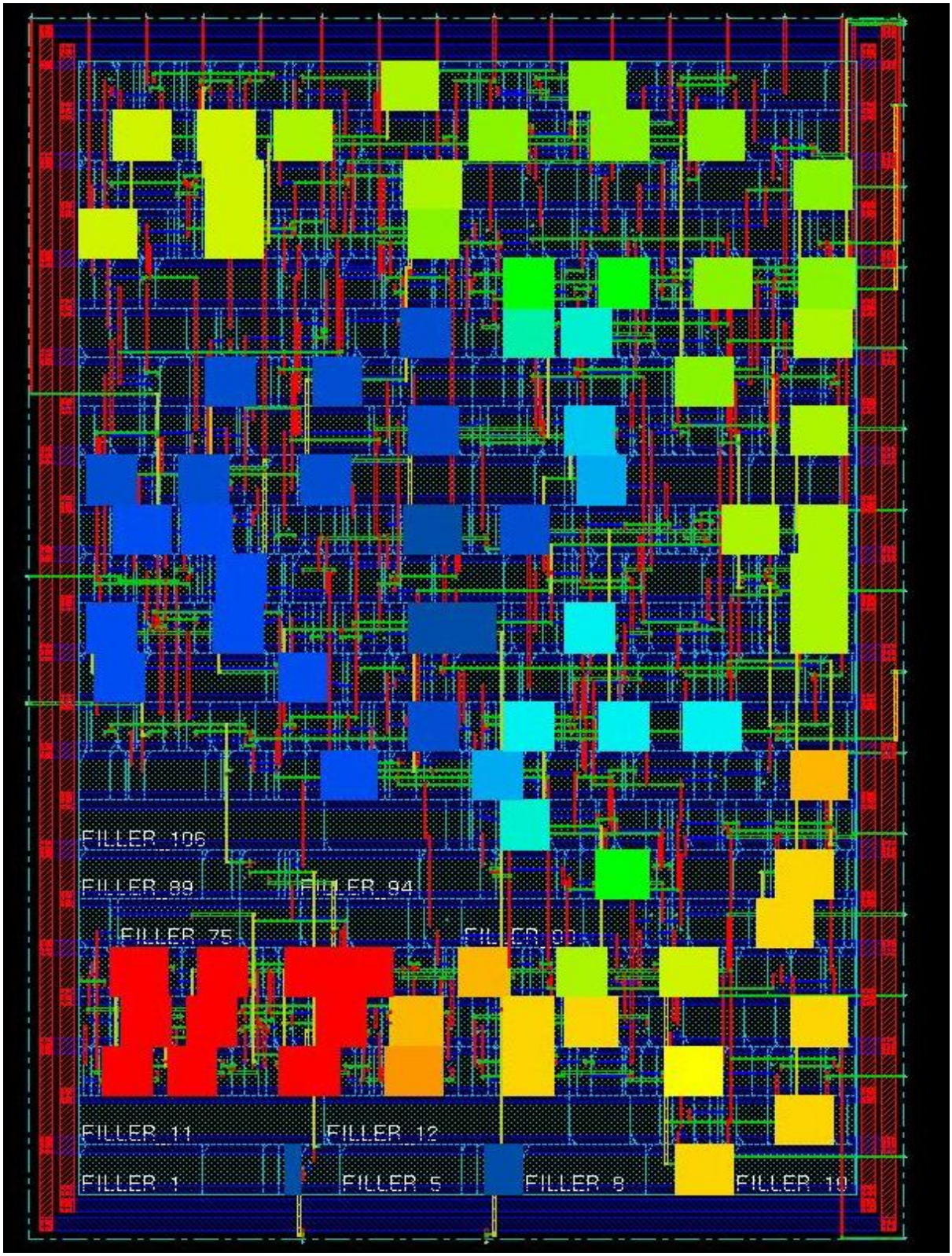


Figure 15: Highlighted clock tree in Encounter

2.7 Routing and Extraction

After CTS the design needs to be routed i.e. the nets are laid using different metal layers. The Routing process can be divided into two steps

i. **Global Routing:** During the global routing the design is divided into different regions called grids and the tool estimates the shortest path within and across different regions without actually laying out the nets on the metal layers.

ii. **Detailed Routing:** During detailed routing the tool uses the information gathered in the Global routing phase to actually lay the nets on the metal layers. The routing tool will route the design while respecting the layout rules such as metal spacing rules, width of different metal layers etc specified in lef files that are input to it.

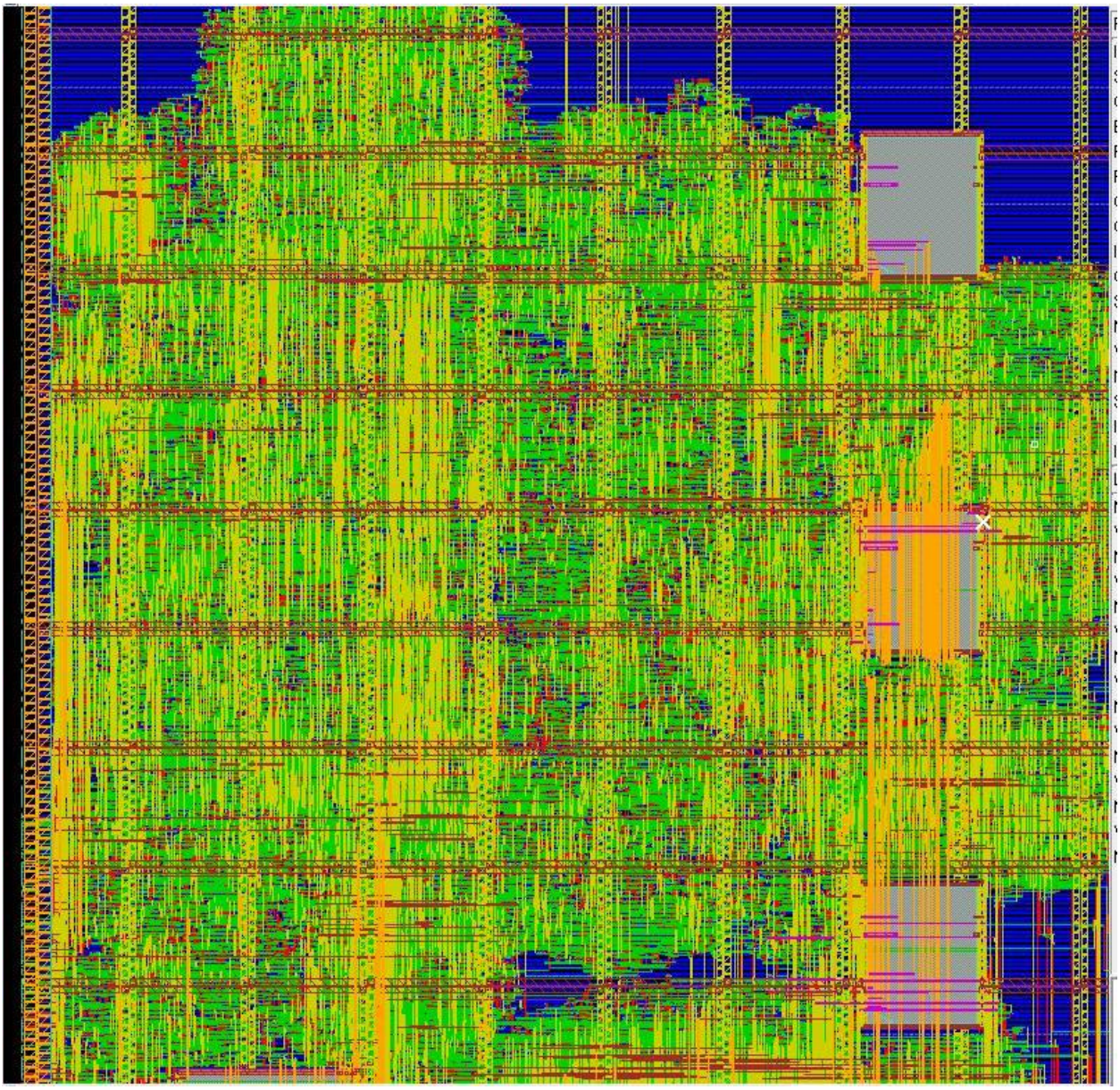


Figure 16: PHY Routing in Encounter

Once the design has been routed it is important to analyze the quality of the routing itself, this can be done by checking for congested spots in the layout. This happens if the routing is performed on a poorly placed design and hence the placement of the design needs to be optimized. Also another important check is to look for any scenic routes

which may occur due to the presence of any highly congested paths in the design, therefore the tool routes the net around these hot spots in order to avoid these hotspots.

```
#
#  Congestion Analysis: (blocked Gcells are excluded from the statistics)
#
#                OverCon       OverCon
#                 #Gcell        #Gcell    %Gcell
#     Layer          (1)           (2)    OverCon
#  ------------------------------------------------
#    Metal 1      2(0.00%)      0(0.00%)   (0.00%)
#    Metal 2      5(0.00%)      1(0.00%)   (0.01%)
#    Metal 3     12(0.01%)      1(0.00%)   (0.01%)
#    Metal 4      0(0.00%)      0(0.00%)   (0.00%)
#    Metal 5      0(0.00%)      0(0.00%)   (0.00%)
#    Metal 6      0(0.00%)      0(0.00%)   (0.00%)
#    Metal 7      0(0.00%)      0(0.00%)   (0.00%)
#  ------------------------------------------------
#     Total     19(0.00%)      2(0.00%)   (0.00%)
#
#  The worst congested Gcell overcon (routing demand over resource in number of tracks) = 2
#
#Complete Global Routing.
#Total wire length = 2737441 um.
#Total half perimeter of net bounding box = 2214580 um.
#Total wire length on LAYER M1 = 69325 um.
#Total wire length on LAYER M2 = 732546 um.
#Total wire length on LAYER M3 = 824031 um.
```

Figure 17: PHY Congestion Analysis table in Encounter

Once the design has been routed the parasitics due to the nets needs to be estimated, which is done during the Extraction process. Once the parasitic contribution of the net is estimated this information is back annotated into the design and the timing of the design is recalculated. Now the paths in the timing reports will have delay contributions due to both cells delays and net delays.

Encounter Command [14]

SetExtractRCMode -detail

extractRC

rcOut -spf <file_name>.spef

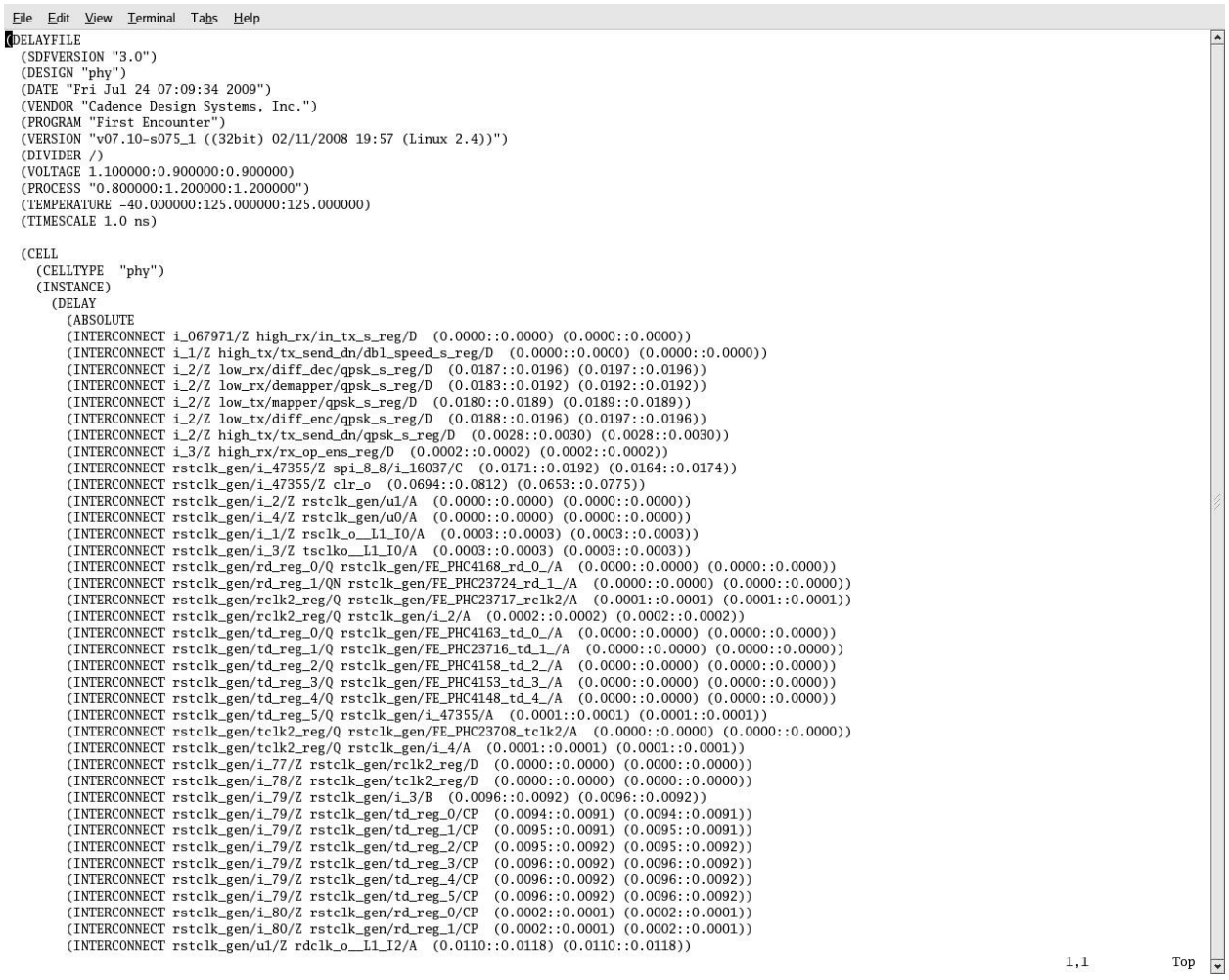
The commonly used formats for parasitic extraction are

i. SPEF : Standard Parasitic Exchange format [17]

```
*|DSPF 1.0
*
*|VENDOR "Silicon Perspective, A Cadence Company"
*|PROGRAM "FirstEncounter System"
*|DIVIDER /
*|DELIMITER :
*|BUSBIT []
*
.SUBCKT complex_multiplier_v1
+ real_result[6] real_result[5] real_result[4] real_result[3] real_result[2] real_result[1] real_result[0] imag_result[6] imag_result[5] imag_result[4]
+ imag_result[3] imag_result[2] imag_result[1] imag_result[0] real_dataa[2] real_dataa[1] real_dataa[0] imag_dataa[2] imag_dataa[1] imag_dataa[0]
+ real_datab[2] real_datab[1] real_datab[0] imag_datab[2] imag_datab[1] imag_datab[0] clock

* Net Section
*
*
*|GROUND_NET 0
*
*|NET add/n169 0.000052PF
*|I (add/clock_r_REG61_S2:Q add/clock_r_REG61_S2 Q O 0.000000PF 49.305
*+ 17.430)
*|I (add/U32:D add/U32 D I 0.001860PF 50.350 16.265)
C1_127 add/U32:D 0 0.000026PF
C2_127 add/clock_r_REG61_S2:Q 0 0.000026PF
R1_127 add/clock_r_REG61_S2:Q add/U32:D 0.653333

*|NET add/n174 0.000028PF
*|I (add/U23:A add/U23 A I 0.003110PF 57.190 24.105)
*|I (add/clock_r_REG46_S2:Q add/clock_r_REG46_S2 Q O 0.000000PF 56.885
*+ 23.750)
C1_133 add/U23:A 0 0.000014PF
C2_133 add/clock_r_REG46_S2:Q 0 0.000014PF
R1_133 add/clock_r_REG46_S2:Q add/U23:A 0.326667

*|NET add/n1 0.000114PF
*|I (add/U4:Z add/U4 Z O 0.000000PF 35.240 23.535)
*|I (add/clock_r_REG49_S2:D add/clock_r_REG49_S2 D I 0.002170PF 34.090
```

Figure 18: Snapshot of SPEF file

ii. SDF : Standard Delay Format [18]



```
File  Edit  View  Terminal  Tabs  Help
(DELAYFILE
  (SDFVERSION "3.0")
  (DESIGN "phy")
  (DATE "Fri Jul 24 07:09:34 2009")
  (VENDOR "Cadence Design Systems, Inc.")
  (PROGRAM "First Encounter")
  (VERSION "v07.10-s075_1 ((32bit) 02/11/2008 19:57 (Linux 2.4))")
  (DIVIDER /)
  (VOLTAGE 1.100000:0.900000:0.900000)
  (PROCESS "0.800000:1.200000:1.200000")
  (TEMPERATURE -40.000000:125.000000:125.000000)
  (TIMESCALE 1.0 ns)

  (CELL
    (CELLTYPE  "phy")
    (INSTANCE)
      (DELAY
        (ABSOLUTE
        (INTERCONNECT i_067971/Z high_rx/in_tx_s_reg/D  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT i_1/Z high_tx/tx_send_dn/dbl_speed_s_reg/D  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT i_2/Z low_rx/diff_dec/qpsk_s_reg/D  (0.0187::0.0196) (0.0197::0.0196))
        (INTERCONNECT i_2/Z low_rx/demapper/qpsk_s_reg/D  (0.0183::0.0192) (0.0192::0.0192))
        (INTERCONNECT i_2/Z low_tx/mapper/qpsk_s_reg/D  (0.0180::0.0189) (0.0189::0.0189))
        (INTERCONNECT i_2/Z low_tx/diff_enc/qpsk_s_reg/D  (0.0188::0.0196) (0.0197::0.0196))
        (INTERCONNECT i_2/Z high_tx/tx_send_dn/qpsk_s_reg/D  (0.0028::0.0030) (0.0028::0.0030))
        (INTERCONNECT i_3/Z high_rx/rx_op_ens_reg/D  (0.0002::0.0002) (0.0002::0.0002))
        (INTERCONNECT rstclk_gen/i_47355/Z spi_8_8/i_16037/C  (0.0171::0.0192) (0.0164::0.0174))
        (INTERCONNECT rstclk_gen/i_47355/Z clr_o  (0.0694::0.0812) (0.0653::0.0775))
        (INTERCONNECT rstclk_gen/i_2/Z rstclk_gen/u1/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/i_4/Z rstclk_gen/u0/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/i_1/Z rsclk_o__L1_IO/A  (0.0003::0.0003) (0.0003::0.0003))
        (INTERCONNECT rstclk_gen/i_3/Z tsclko__L1_IO/A  (0.0003::0.0003) (0.0003::0.0003))
        (INTERCONNECT rstclk_gen/rd_reg_0/Q rstclk_gen/FE_PHC4168_rd_0_/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/rd_reg_1/QN rstclk_gen/FE_PHC23724_rd_1_/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/rclk2_reg/Q rstclk_gen/FE_PHC23717_rclk2/A  (0.0001::0.0001) (0.0001::0.0001))
        (INTERCONNECT rstclk_gen/rclk2_reg/Q rstclk_gen/i_2/A  (0.0002::0.0002) (0.0002::0.0002))
        (INTERCONNECT rstclk_gen/td_reg_0/Q rstclk_gen/FE_PHC4163_td_0_/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/td_reg_1/Q rstclk_gen/FE_PHC23716_td_1_/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/td_reg_2/Q rstclk_gen/FE_PHC4158_td_2_/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/td_reg_3/Q rstclk_gen/FE_PHC4153_td_3_/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/td_reg_4/Q rstclk_gen/FE_PHC4148_td_4_/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/td_reg_5/Q rstclk_gen/i_47355/A  (0.0001::0.0001) (0.0001::0.0001))
        (INTERCONNECT rstclk_gen/tclk2_reg/Q rstclk_gen/FE_PHC23708_tclk2/A  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/tclk2_reg/Q rstclk_gen/i_4/A  (0.0001::0.0001) (0.0001::0.0001))
        (INTERCONNECT rstclk_gen/i_77/Z rstclk_gen/rclk2_reg/D  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/i_78/Z rstclk_gen/tclk2_reg/D  (0.0000::0.0000) (0.0000::0.0000))
        (INTERCONNECT rstclk_gen/i_79/Z rstclk_gen/i_3/B  (0.0096::0.0092) (0.0096::0.0092))
        (INTERCONNECT rstclk_gen/i_79/Z rstclk_gen/td_reg_0/CP  (0.0094::0.0091) (0.0094::0.0091))
        (INTERCONNECT rstclk_gen/i_79/Z rstclk_gen/td_reg_1/CP  (0.0095::0.0091) (0.0095::0.0091))
        (INTERCONNECT rstclk_gen/i_79/Z rstclk_gen/td_reg_2/CP  (0.0095::0.0092) (0.0095::0.0092))
        (INTERCONNECT rstclk_gen/i_79/Z rstclk_gen/td_reg_3/CP  (0.0096::0.0092) (0.0096::0.0092))
        (INTERCONNECT rstclk_gen/i_79/Z rstclk_gen/td_reg_4/CP  (0.0096::0.0092) (0.0096::0.0092))
        (INTERCONNECT rstclk_gen/i_79/Z rstclk_gen/td_reg_5/CP  (0.0096::0.0092) (0.0096::0.0092))
        (INTERCONNECT rstclk_gen/i_80/Z rstclk_gen/rd_reg_0/CP  (0.0002::0.0001) (0.0002::0.0001))
        (INTERCONNECT rstclk_gen/i_80/Z rstclk_gen/rd_reg_1/CP  (0.0002::0.0001) (0.0002::0.0001))
        (INTERCONNECT rstclk_gen/u1/Z rdclk_o__L1_I2/A  (0.0110::0.0118) (0.0110::0.0118))
                                                                                    1,1           Top
```

Figure 19: Snapshot of SDF file

2.8 Static Timing Analysis (STA)

Static Timing Analysis is technique used to check if the design meets the timing requirements. Static Timing Analysis is a preferred method to do timing analysis as

opposed to Dynamic simulation because STA requires considerably lower runtime than dynamic simulation and moreover the completeness of the check by dynamic simulation is dependent on the coverage of the test vectors that are used in the dynamic simulation.

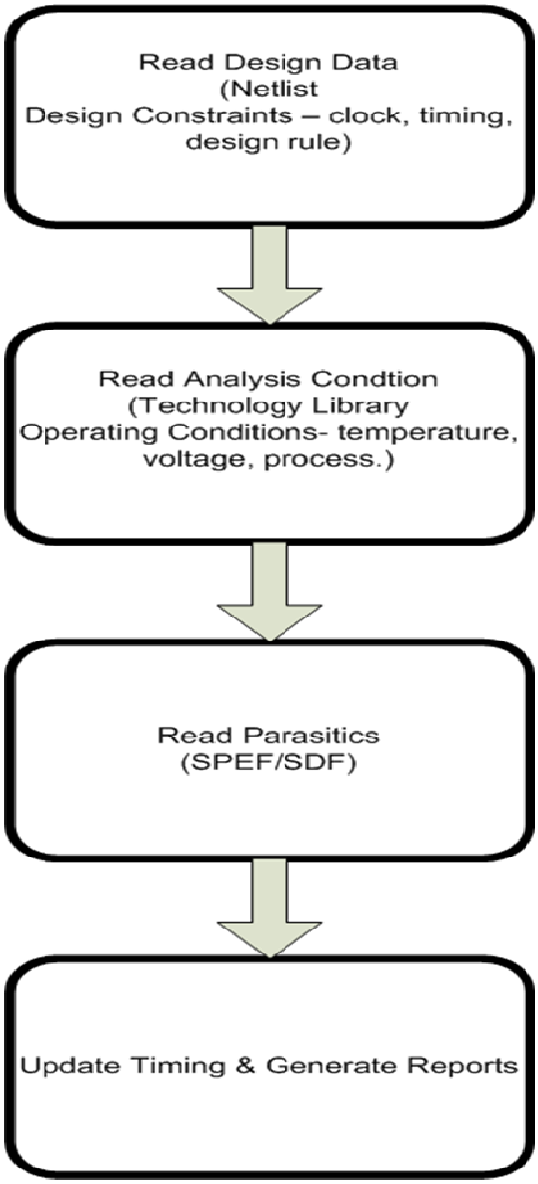


Figure 20: Static Timing Analysis Block Diagram

STA tools such as Prime Time [19] (PT) can be used to back annotate the parasitics onto the placed design and the timing can be analyzed. PT supports different analysis modes

such as worst case – best case: use the max library to analyze setup and min library to analyze hold, this way it is possible to analyze the design in the worst case operating condition possible.

The types of paths analyzed using STA can be broadly classified into 4 categories

i. Start point : Primary inputs ; Endpoints : Data pin of all flops

   PT Command: *report_timing –from [all_inputs] –to [all_registers –data_pins]*

ii. Start point : Primary inputs ; Endpoints : Primary outputs

   PT Command: *report_timing –from [all_inputs] -to [all_outputs]*

iii. Start point : Output of flops ; Endpoints : Data pin of all flops

   PT Command: *report_timing –from [all_registers –clock_pins] –to [all_registers –data_pins]*

iv. Start point : Output of flops ; Endpoints : Primary Outputs

   PT Command: *report_timing –from [all_registers -clock_pins] –to [all_outputs]*

Once the appropriate timing reports are generated and analyzed the timing violations that still remain need to fixed. This can be done by iterating with better timing driven placement and if the timing violations are massive then it might require a complete re-floor planning to optimize the critical paths.

```
Startpoint: spi_8_8/cdout_reg_0
            (rising edge-triggered flip-flop clocked by phy_sclk)
Endpoint: spi_8_8/pdin_reg_reg_223
            (recovery check against rising-edge clock phy_sclk)
Path Group: **async_default**
Path Type: max

Point                                          Fanout       Incr       Path
-------------------------------------------------------------------------------
clock phy_sclk (rise edge)                                  0.00       0.00
clock network delay (ideal)                                 0.30       0.30
spi_8_8/cdout_reg_0/CP (FD2QSVTX1)                          0.00       0.30 r
spi_8_8/cdout_reg_0/Q (FD2QSVTX1)                           0.22 *     0.52 f
spi_8_8/FE_PHN18938_cdout_0___ac__ (net)       1
spi_8_8/FE_PHC18938_cdout_0___ac__/A (BFHVTX1)              0.00 *     0.52 f
spi_8_8/FE_PHC18938_cdout_0___ac__/Z (BFHVTX1)              0.14 *     0.67 f
spi_8_8/FE_PHN18937_cdout_0___ac__ (net)       1
spi_8_8/FE_PHC18937_cdout_0___ac__/A (BFHVTX1)              0.00 *     0.67 f
spi_8_8/FE_PHC18937_cdout_0___ac__/Z (BFHVTX1)              0.14 *     0.80 f
spi_8_8/FE_PHN18936_cdout_0___ac__ (net)       1
spi_8_8/FE_PHC18936_cdout_0___ac__/A (BFHVTX1)              0.00 *     0.80 f
spi_8_8/FE_PHC18936_cdout_0___ac__/Z (BFHVTX1)              0.18 *     0.98 f
spi_8_8/cdout[0]__ac__ (net)                   2
spi_8_8/i_16037/B (AO6CLVTX4)                               0.00 *     0.98 f
spi_8_8/i_16037/Z (AO6CLVTX4)                               0.27 *     1.25 r
spi_8_8/FE_OFN218_clr2 (net)                  56
spi_8_8/FE_OFC219_clr2/A (BFSVTX2)                          0.01 *     1.26 r
spi_8_8/FE_OFC219_clr2/Z (BFSVTX2)                          0.38 *     1.63 r
spi_8_8/FE_OFN220_clr2 (net)                  41
spi_8_8/FE_OFC221_clr2/A (BFSVTX4)                          0.00 *     1.64 r
spi_8_8/FE_OFC221_clr2/Z (BFSVTX4)                          0.34 *     1.98 r
spi_8_8/FE_OFN222_clr2 (net)                  75
spi_8_8/FE_OFC223_clr2/A (BFSVTX4)                          0.01 *     1.98 r
spi_8_8/FE_OFC223_clr2/Z (BFSVTX4)                          0.33 *     2.31 r
spi_8_8/FE_OFN224_clr2 (net)                  71
spi_8_8/FE_OFC224_clr2/A (BFSVTX4)                          0.02 *     2.34 r
spi_8_8/FE_OFC224_clr2/Z (BFSVTX4)                          0.34 *     2.67 r
spi_8_8/reset_n_out (net)                     75
spi_8_8/pdin_reg_reg_223/CD (FD2QSVTX1)                     0.03 *     2.70 r
data arrival time                                                      2.70

clock phy_sclk (rise edge)                                 16.00      16.00
clock network delay (ideal)                                 0.00      16.00
clock uncertainty                                          -0.30      15.70
spi_8_8/pdin_reg_reg_223/CP (FD2QSVTX1)                                15.70 r
library recovery time                                      -0.17 *    15.53
data required time                                                    15.53
-------------------------------------------------------------------------------
data required time                                                    15.53
data arrival time                                                     -2.70
-------------------------------------------------------------------------------
slack (MET)                                                           12.84
```

Figure 21: Timing Report in Prime Time

2.9 Power Estimation

In case of designs that have a strict power budget, it is important to estimate and optimize the power within the power budget. Power estimation can be carried in tools such as Power Compiler [20]. In order to estimated power consumption of the design it is essential to have the switching activity information for the design for a given input test vector. The switching activity information for the design can be obtained by conducting a dynamic simulation of the design for a set or individual test vector that activates a substantial part of the design. The dynamic simulation is carried out in Modelsim [10] and the corresponding VCD (value change dump) file is generated. This VCD file is then converted to SAIF (switching activity information file) [21] in Power compiler.

The inputs for power estimation are as follows

i. Post routed Netlist.
ii. Extracted Parasitic information
iii. Design constraints in form of sdc file
iv. Switching activity information about the various nodes in the design.

Once the above mentioned inputs are read into Power Compiler, the power report can be generated using the Power Compiler command **report_power.**

Hierarchy	Switch Power	Int Power	Leak Power	Total Power	%
phy	0.351	0.711	7.64e+10	77.490	100.0
high_rx (phy_high_rx)	0.130	0.312	2.97e+10	30.106	38.9
rx_pass_up (rx_pass_up)	4.43e-04	1.79e-02	1.79e+09	1.808	2.3
i_18806 (AWDP_LE_17)	0.000	0.000	1.42e+07	1.42e-02	0.0
i_18813 (AWDP_DEC_1501676)	0.000	0.000	1.69e+07	1.69e-02	0.0
i_18818 (AWDP_INC_14)	0.000	0.000	1.80e+07	1.80e-02	0.0
i_18823 (AWDP_SUB_19)	0.000	0.000	3.28e+07	3.28e-02	0.0
i_18833 (AWDP_INC_4501757)	0.000	0.000	1.42e+07	1.42e-02	0.0
pld_dscr (dscr32p)	0.000	3.86e-03	3.26e+08	0.330	0.4
par_pad_off (par_pad_off)	0.000	3.49e-03	2.89e+08	0.292	0.4
i_199822 (AWDP_SUB_7498225)	0.000	0.000	3.28e+07	3.28e-02	0.0
rsd4p (rsd4p)	0.130	0.224	2.03e+10	20.677	26.7
rsd4p_sr126x32 (rsd4p_sr126x32)	5.50e-02	5.73e-02	3.43e+08	0.455	0.6
rsd4p_ecb (rsd4p_ecb)	7.49e-02	7.57e-02	6.68e+09	6.835	8.8
rsd4p_chien (rsd4p_chien)	4.23e-03	3.21e-02	3.52e+09	3.556	4.6
rsd4p_erpt (rsd4p_erpt)	0.000	7.11e-04	1.14e+08	0.115	0.1
rsd4p_ecc8 (rsd4p_ecc8)	0.000	1.16e-03	1.97e+08	0.199	0.3
rsd4p_ecc0_0 (rsd4p_ecc0_0)	0.000	1.17e-03	1.02e+08	0.103	0.1
rsd4p_ecc2_0 (rsd4p_ecc2_0)	0.000	1.18e-03	2.25e+08	0.227	0.3
rsd4p_ecc4_0 (rsd4p_ecc4_0)	0.000	1.15e-03	2.35e+08	0.237	0.3
rsd4p_ecc6_0 (rsd4p_ecc6_0)	0.000	1.16e-03	2.18e+08	0.219	0.3
rsd4p_ecc1 (rsd4p_ecc1_0)	0.000	1.17e-03	1.27e+08	0.128	0.2
rsd4p_ecc0_1 (rsd4p_ecc0_1)	0.000	1.17e-03	1.02e+08	0.103	0.1
rsd4p_ecc2_1 (rsd4p_ecc2_1)	0.000	1.16e-03	2.44e+08	0.245	0.3
rsd4p_ecc4_1 (rsd4p_ecc4_1)	0.000	1.16e-03	2.44e+08	0.245	0.3
rsd4p_ecc6_1 (rsd4p_ecc6_1)	0.000	1.17e-03	2.41e+08	0.242	0.3
rsd4p_gfmul_pipe2_a (rsd4p_gfmul_pipe2_0)	0.000	8.45e-04	1.42e+08	0.143	0.2
rsd4p_gfmul_pipe2_b (rsd4p_gfmul_pipe2_1)	0.000	8.41e-04	1.41e+08	0.142	0.2
rsd4p_gfmul_pipe2_c (rsd4p_gfmul_pipe2_2)	0.000	8.50e-04	1.21e+08	0.122	0.2
rsd4p_gfmul_pipe2_d (rsd4p_gfmul_pipe2_3)	0.000	8.39e-04	1.42e+08	0.143	0.2
rsd4p_forney (rsd4p_forney)	7.63e-02	4.37e-02	3.16e+09	3.276	4.2
rsd4p_ecc7 (rsd4p_ecc7)	0.000	1.19e-03	2.55e+08	0.256	0.3
rsd4p_ecc5 (rsd4p_ecc5)	0.000	1.19e-03	2.29e+08	0.230	0.3
rsd4p_ecc3 (rsd4p_ecc3)	0.000	1.18e-03	2.35e+08	0.236	0.3
rsd4p_ecc0 (rsd4p_ecc0_2)	N/A	4.70e-03	1.02e+08	9.96e-02	0.1
rsd4p_ecc1 (rsd4p_ecc1_1)	N/A	1.62e-03	1.84e+08	0.179	0.2
rsd4p_ecc2 (rsd4p_ecc2_2)	N/A	7.34e-03	2.38e+08	0.238	0.3
rsd4p_ecc4 (rsd4p_ecc4_2)	0.000	1.18e-03	2.31e+08	0.233	0.3
rsd4p_ecc6 (rsd4p_ecc6_2)	0.000	1.20e-03	2.33e+08	0.234	0.3
rsd4p_gfmul_pipe2_a (rsd4p_gfmul_pipe2_4)	1.15e-02	8.48e-04	1.26e+08	0.138	0.2

Figure 22: Hierarchical Power Report in Prime Power

2.10 DRC, LVS and Design Signoff

The main objective of DRC (Design Rule Check) [32] is to make sure that the design layout conforms to the set of Design Rules specified in the technology file of the target process. The motive behind these rules is to improve the yield and reliability of the design. Any violation of the Design rules may result in the design being non functional.

Some typical design rule checks in ASIC design are as follows.

1. **Spacing Rules:** Metal to Metal Spacing, Active to active spacing, well to well spacing etc.
2. **Minimum Channel length of the device.**
3. **Minimum metal width.**
4. **Metal fill Density.**

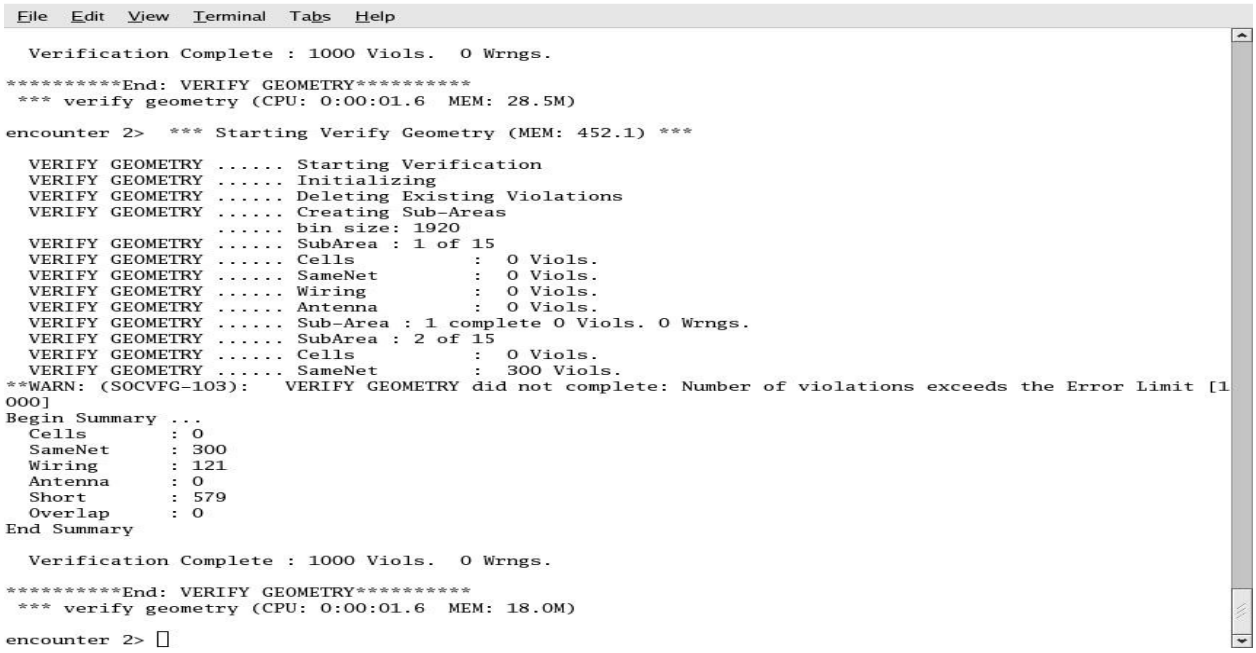


Figure 23: DRC report in Encounter

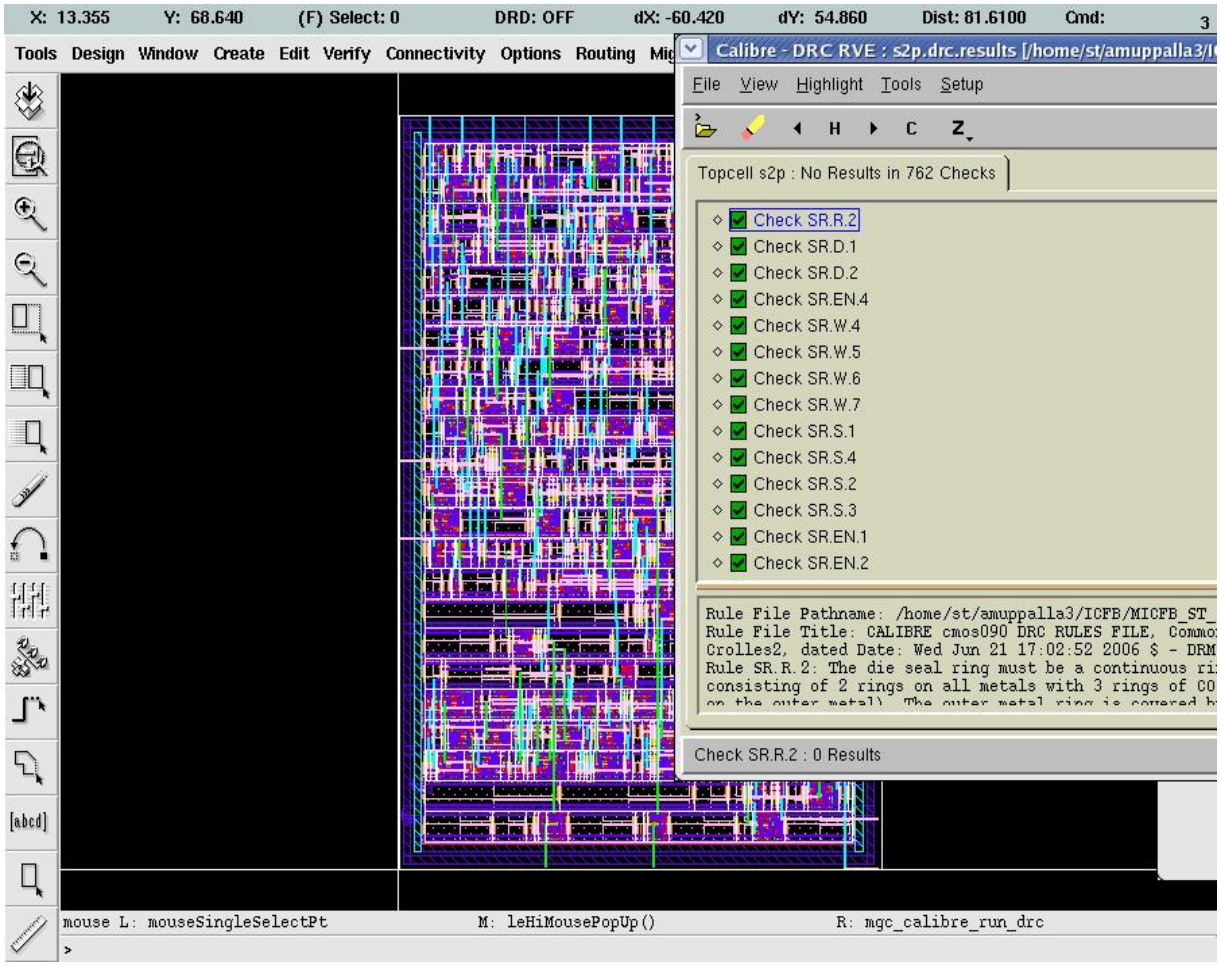


Figure 24: DRC report in Calibre

In addition to DRC rules in sub nanometer nodes more rules known as Design for Manufacturability (DFM) [22] need to be satisfied in order to improve functional yield, parametric yield and reliability.

A design that meets all the required design rules does not necessarily represent the circuit intended to be fabricated. Another check known as Layout vs. Schematic (LVS) [23] is

required to ensure that the drawn shapes in the layout represents the desired electrical components in the design and also the connection between these components are as intended.

Typically LVS check involves the following steps [23]

1. **Extraction:** In this step the LVS tool runs through the layout to identify what components are represented and how they are connected to other components.
2. **Reduction:** In this step the extracted components are combined as defined in the layout and a netlist representation of the layout is generated.
3. **Comparison:** In this step the extracted layout netlist is compared with the netlist from the circuit schematic.

Typical violations that are encountered during LVS are as follows.

1. **Shorts:** Two or more wires are connected when they are not supposed to be.
2. **Open:** Wires are not connected or partially connected when they are supposed to be connected.
3. **Missing component:** A component in the schematic is missing from the layout.
4. **Component Mismatches:** Components of incorrect type are used in the layout.
5. **Property Errors:** The size of the component used in the layout is different.

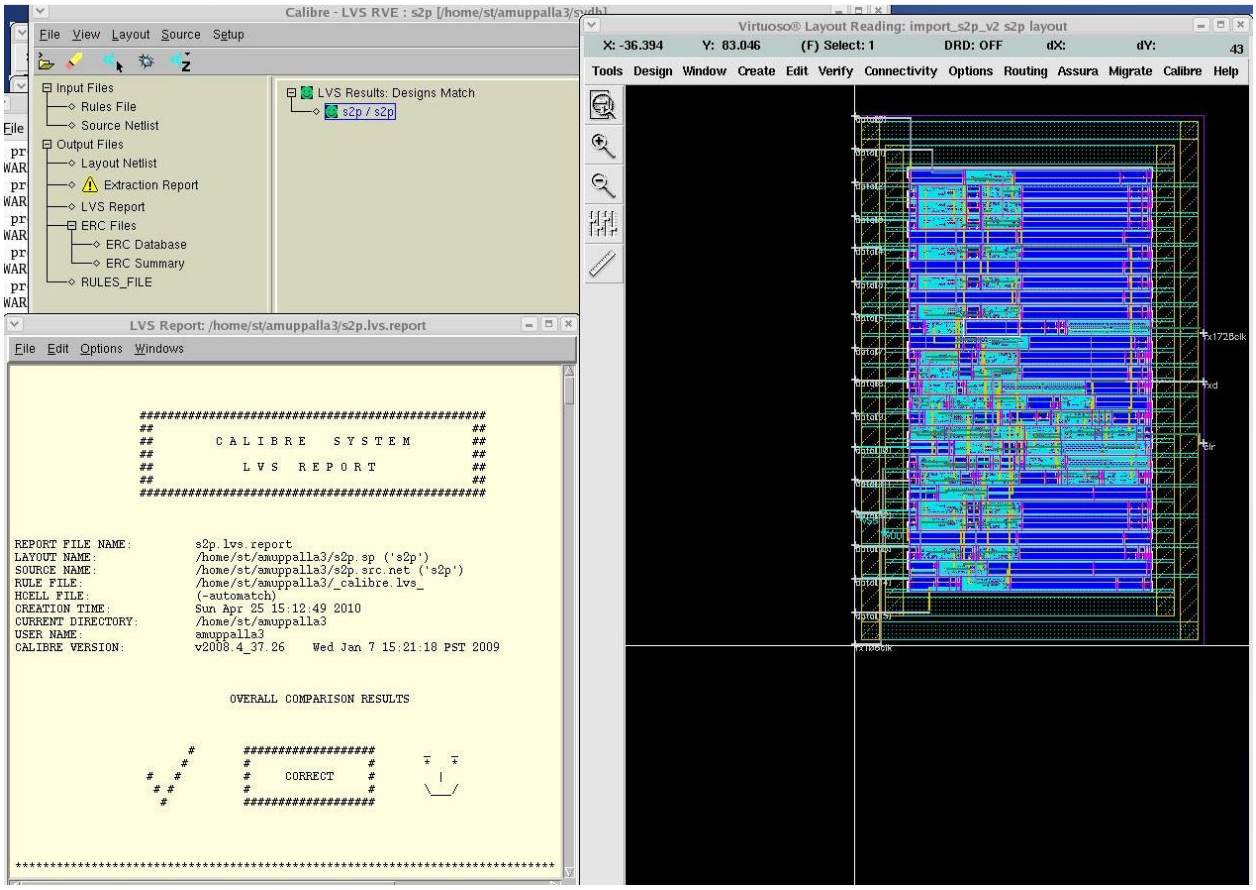


Figure 25: LVS report in Calibre

Once the design has satisfied the timing requirements, DRC and LVS checks if the design is ready for signoff. At this stage the design is extracted in GDSII format [24] which is a binary file format for planar geometric shapes, text labels and other layout information and delivered for fabrication.

3 STANDARD CELL DESIGN

A Standard cell library is a collection of basic logic functions which can be used to implement more complicated logic functions. The library consists of cells whose layout is of a fixed height and variable width. As a result of their fixed height they can be placed with ease in rows by design automation tools. The library generally consists of variable drive strengths of a particular logic function, for example in case of an nand gate the library could have drive strengths of X1, X2, X5, X10 etc in order to satisfy different drive strength requirements that may arise depending on the load the logic function is used to drive. A standard cell library typically consists of timing information which indicates propagation delay, rise time, fall time etc and layout information such as position of cell pins, cell width etc.

One of the basics steps in the physical design of standard cells is the determination of the horizontal and vertical track information. These tracks act as guides for auto place and route tools while placing and establishing connections between the different standard cells. The track information is typically estimated using the spacing and width information of the first two metal layers. Once the track information has been established, the height and width of the standard cells needs to be fixed. The height of the standard cells is typically fixed to be an integral multiple of the track. The height of the standard cell is maintained throughout the library and the width of the standard cells varies depending on the drive strength and logic function of the standard cell [33].

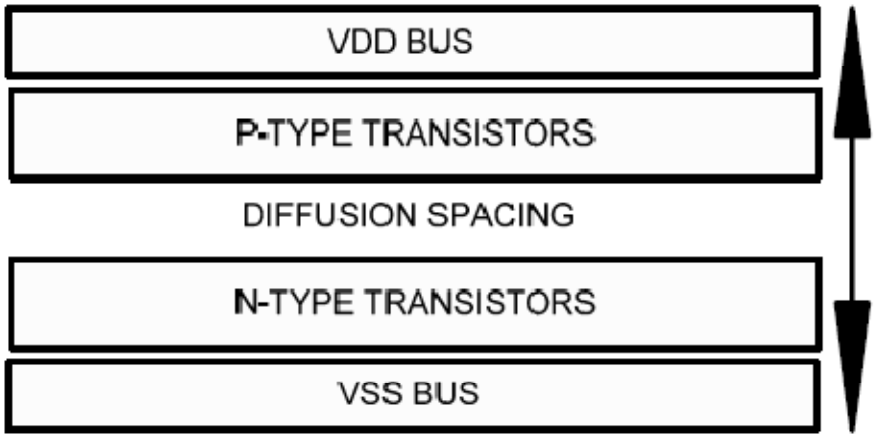


Figure 26: Height of general standard cell

Table 1: Standard cell height for 45nm and 90nm

Technology	Standard cell height
45nm	3.2µm
90nm	4.3µm

Some important guidelines that need to be followed during the layout of the standard cells are: Firstly, it is preferable to use the first conducting layer for all the routing within the standard cell, and if the second routing layer is required, its use should be kept to a minimum. This is done to ensure that most of the remaining layers are available for signal routing. Secondly, all the internal node capacitances should be kept to a minimum and must be kept close to the VDD or GND, this helps in minimizing body effect i.e. the change in device threshold voltage when the source to bias voltage changes [33].

The Standard cell library has to be modeled and characterized for different operating conditions so that they can be used in various applications. The operating conditions are as follows:

Table 2: Operating Conditions

Corner	Temp (℃)	Volt (V)
Nominal	25	1
Worst Case	125	0.9
Best Case	-40	1.1

The following characteristics of the basic standard cells are measured:

Rise time: It is defined as the time taken by the output to go from 10%VDD to 90%VDD.

Fall time: It is defined as the time taken by the output to go from 90%VDD to 10%VDD.

Delay: It is defined as the time difference between the 50% point of the input and the output.

Setup time: (Applies to Flip Flops) It is defined as the time before the clock transition for which the D input should be stable for the value to be correctly latched.

Hold time: (Applies to Flip Flops) It is defined as the time after the clock transition for which the D input should be stable for the value to be correctly latched.

Note: All the inputs to the standard cells have a rise and fall time of 35ps.

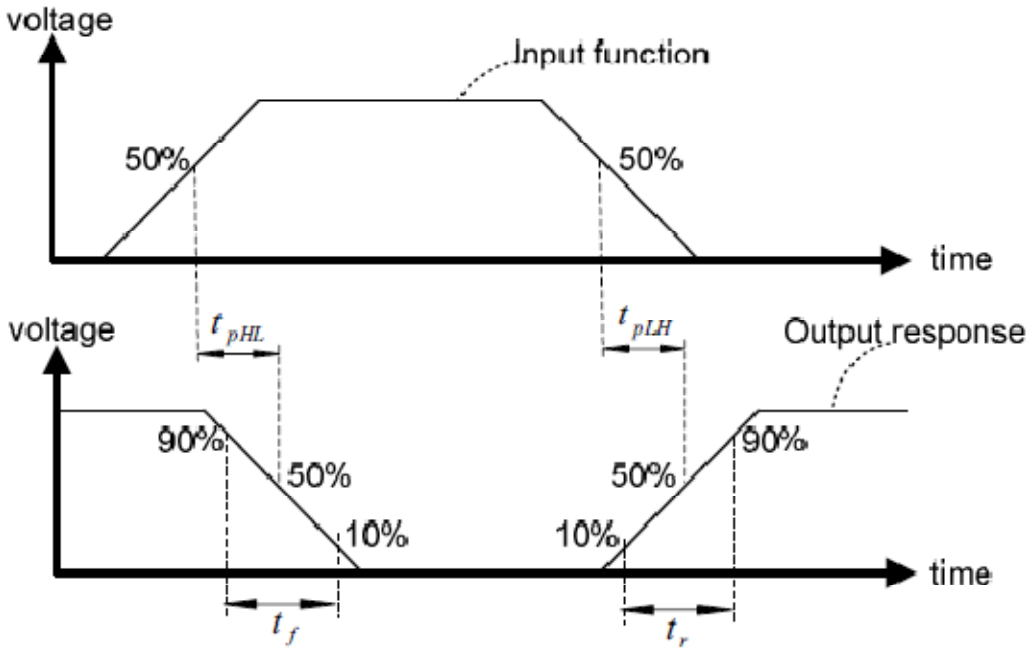


Figure 27: Timing parameters for standard cells

3.1 Inverter

Figure 28: Inverter Symbol

Table 3: Inverter Truth Table

A	Z
1	0
0	1

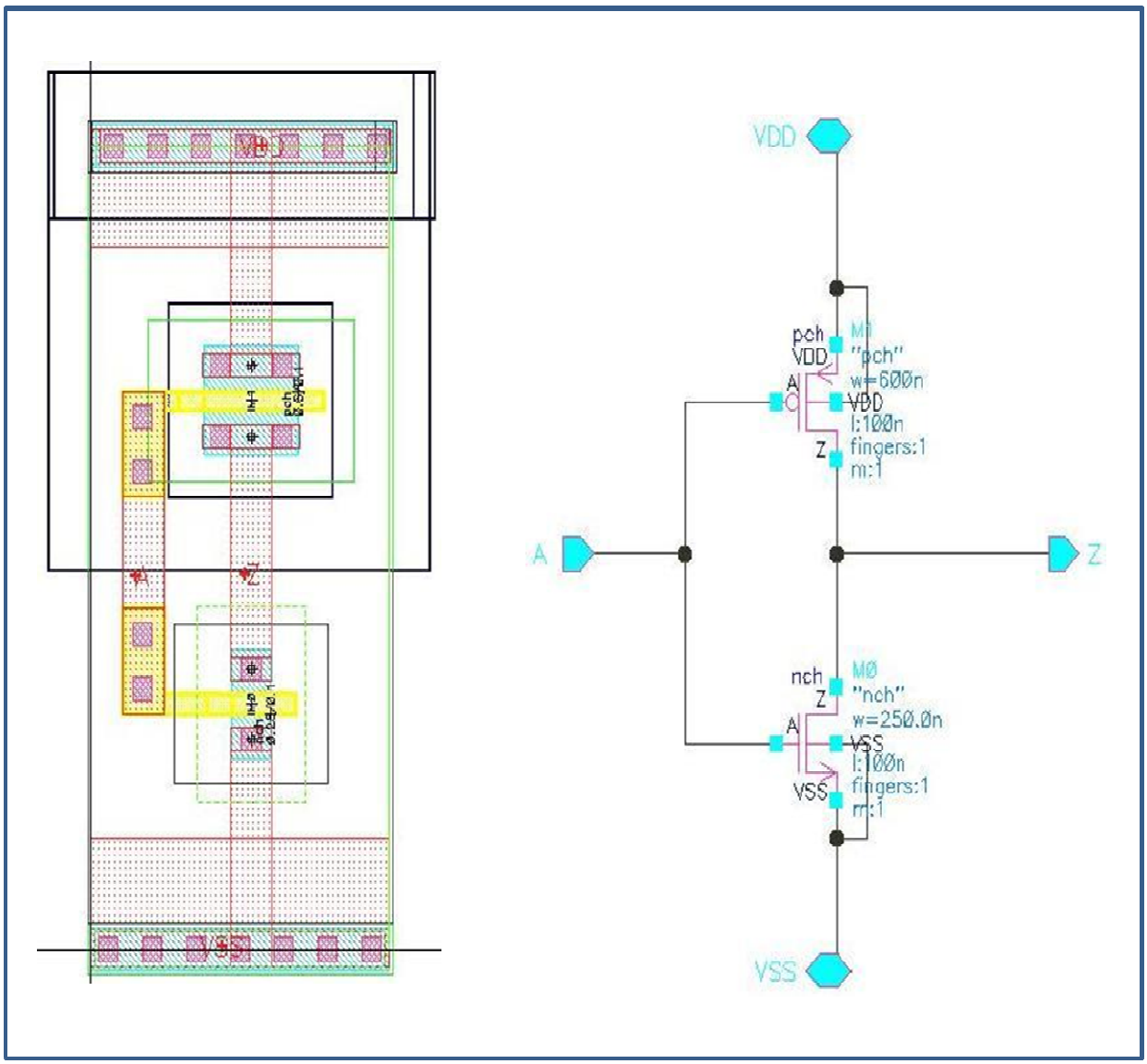


Figure 29: 90nm Inverter Schematic and Layout

Table 4: Inverter delay characteristics at 90nm

	Rise Time	Fall Time	Delay	Av Power
Worst Case	16.8ps	13.96ps	14.65ps	1.76uW
Nominal	14.50ps	11.22ps	13.19ps	1.951uW
Best Case	13.01ps	9.95ps	11.54ps	2.186uW

Area = 4.3μm * 2.19 μm = 9.417μm^2

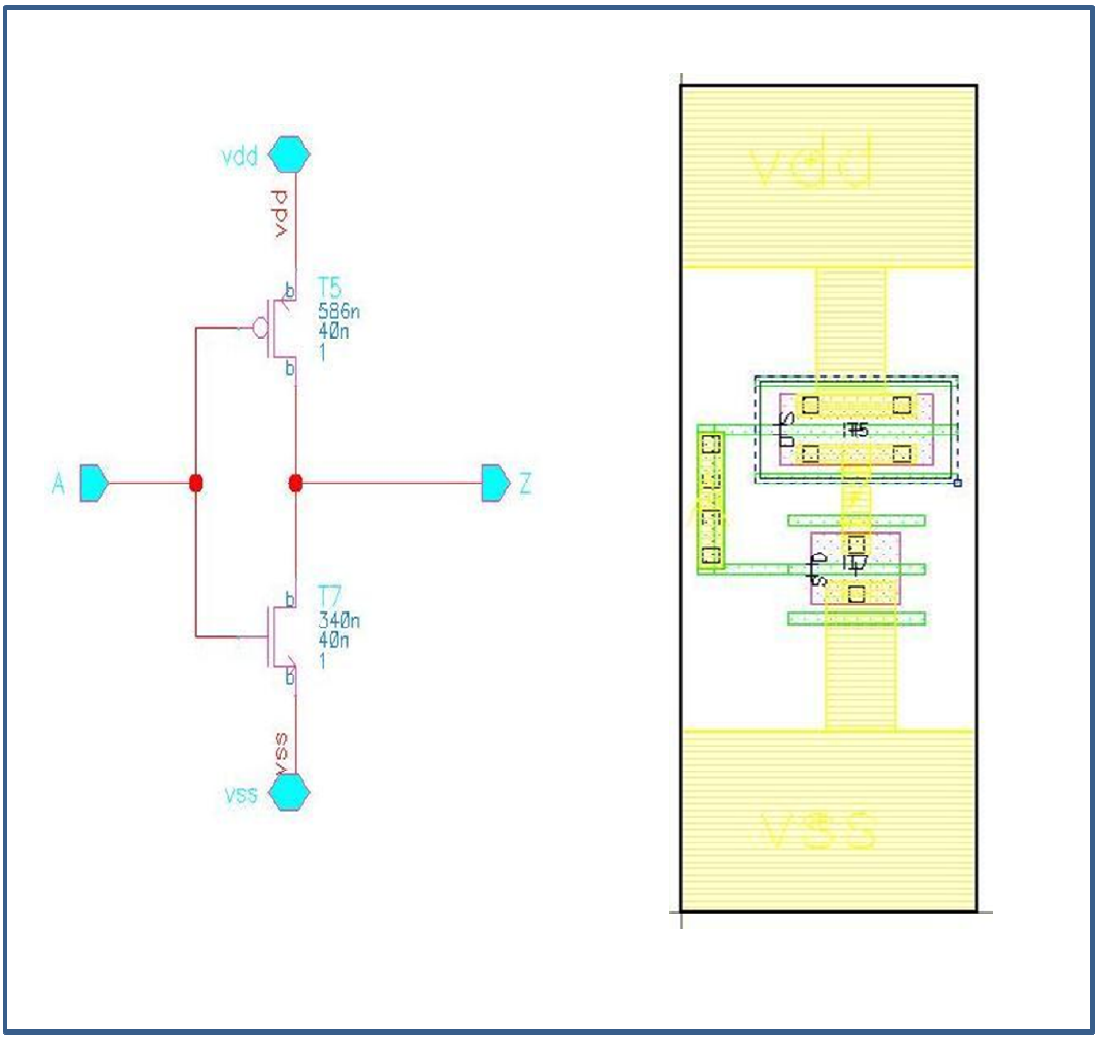


Figure 30: 45nm Inverter Schematic and Layout

Table 5: Inverter delay characteristics at 45nm

	Rise Time	Fall Time	Delay	Av Power
Worst Case	8.28ps	8.28ps	1.97ps	1.77uW
Nominal	8.01ps	7.83ps	0.8ps	2.49uW
Best Case	7.686ps	7.47ps	0.44ps	3.17uW

Area = 3.2μm * 1.126 μm = 3.60μm^2

3.2 NAND

Figure 31 : NAND Symbol

Table 6 : NAND Truth Table

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

Table 32: NAND delay characteristics at 90nm

	Rise Time	Fall Time	Delay	Total Power
Worst Case	38.6ps	75.2ps	51.6ps	3.555uW
Nominal	31.1ps	57.02ps	38.95ps	4.24uW
Best Case	22.74ps	41.8ps	31.38ps	4.457uW

Area = 4.3μm * 3.06 μm = 13.15μm^2

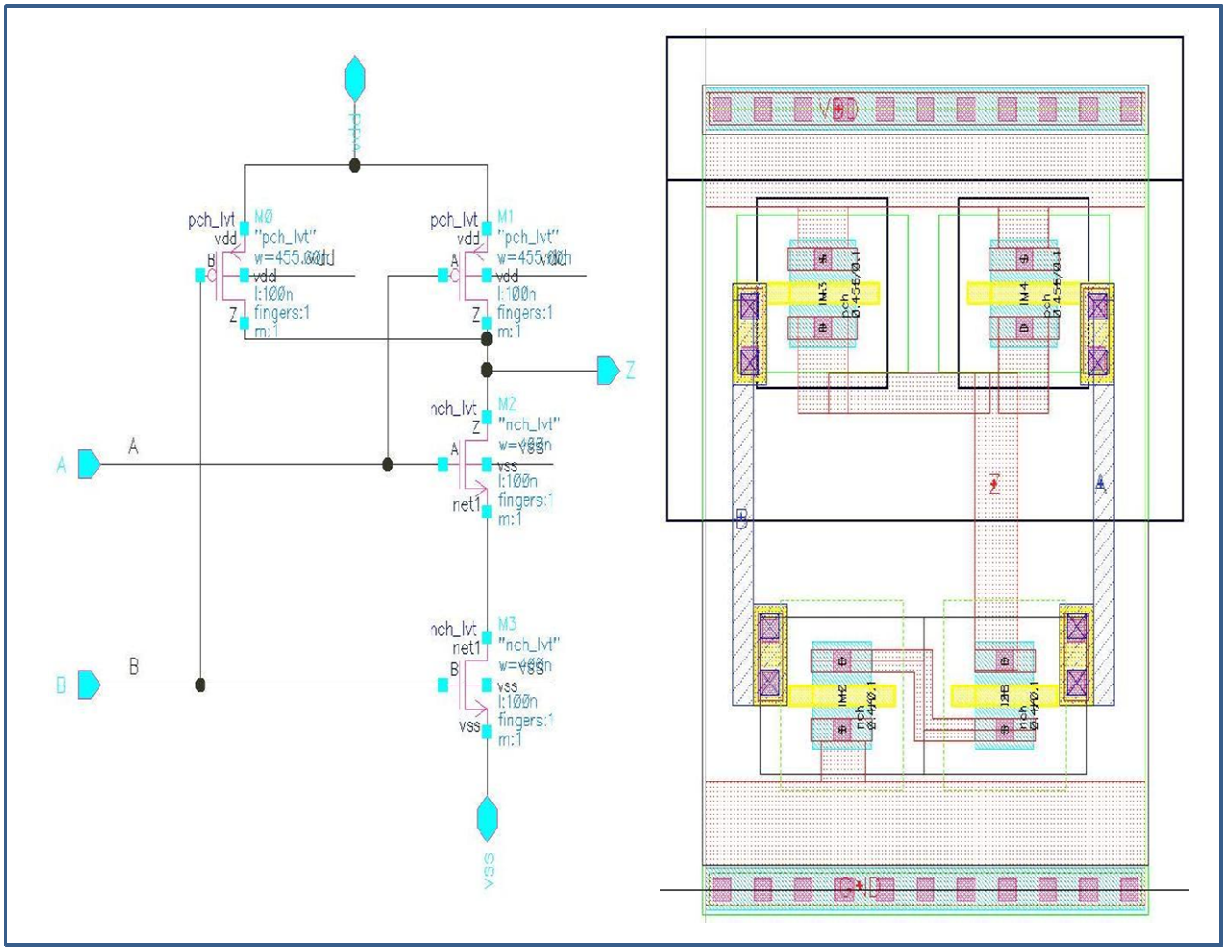


Figure 33 : 90nm NAND schematic and Layout

Table 34: NAND delay characteristics at 45nm

	Rise Time	Fall Time	Delay	Total Power
Worst Case	10.9ps	8.70ps	8.27ps	2.85uW
Nominal	10.12ps	8.42ps	3.14ps	3.17uW
Best Case	9.93ps	7.68ps	2.489ps	3.9uW

Area = 3.2μm * 1.933 μm = 6.18μm^2

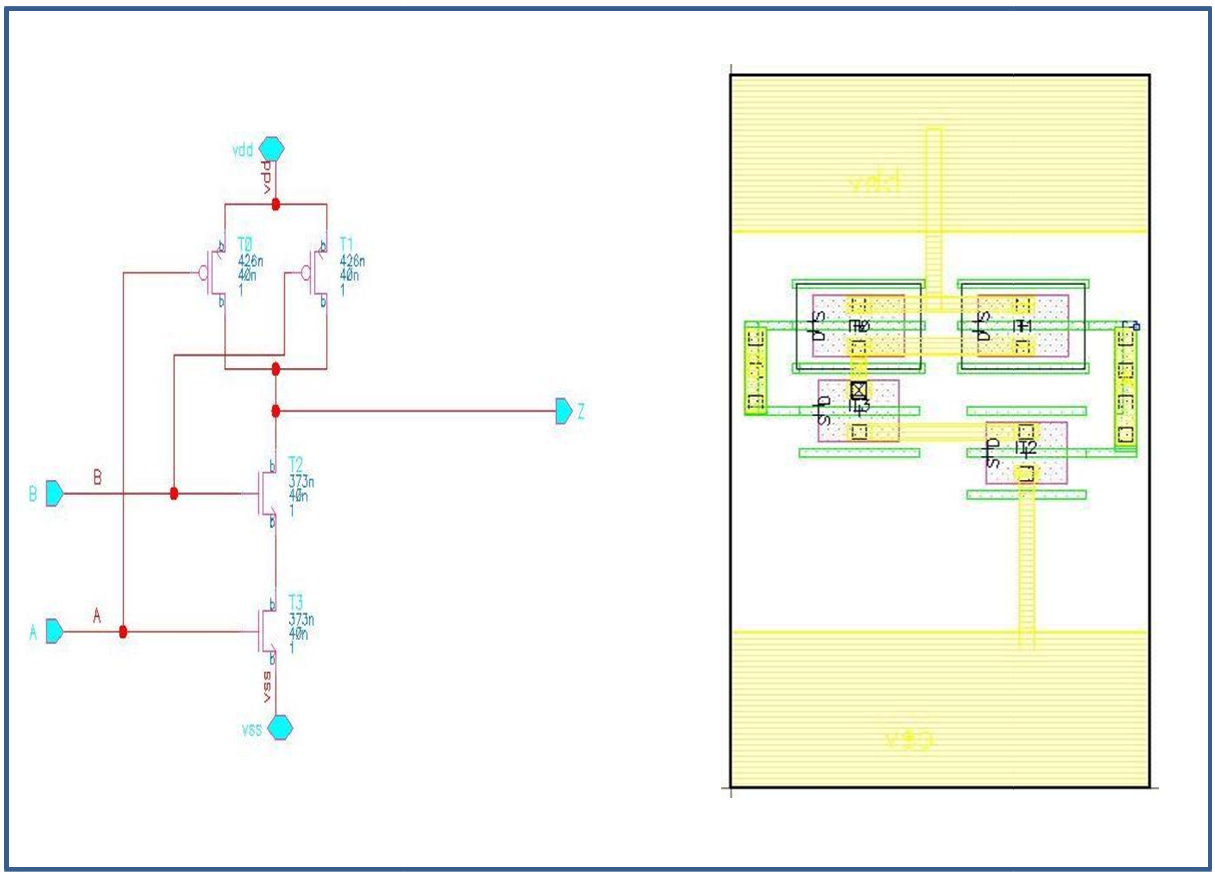


Figure 35: 45nm NAND schematic and Layout

3.3 NOR

Figure 36: NOR Symbol

Table 7: NOR Truth Table

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

Table 8: NOR delay characteristics at 90nm

	Rise Time	Fall Time	Delay	Total Power
Worst Case	116.75ps	53.71ps	51.8ps	9.1uW
Nominal	92.7ps	36.9ps	44.8ps	9.978uW
Best Case	74.06ps	28.8ps	38.05ps	11.36uW

Area = 4.3μm * 2.78 μm = 11.94μm^2

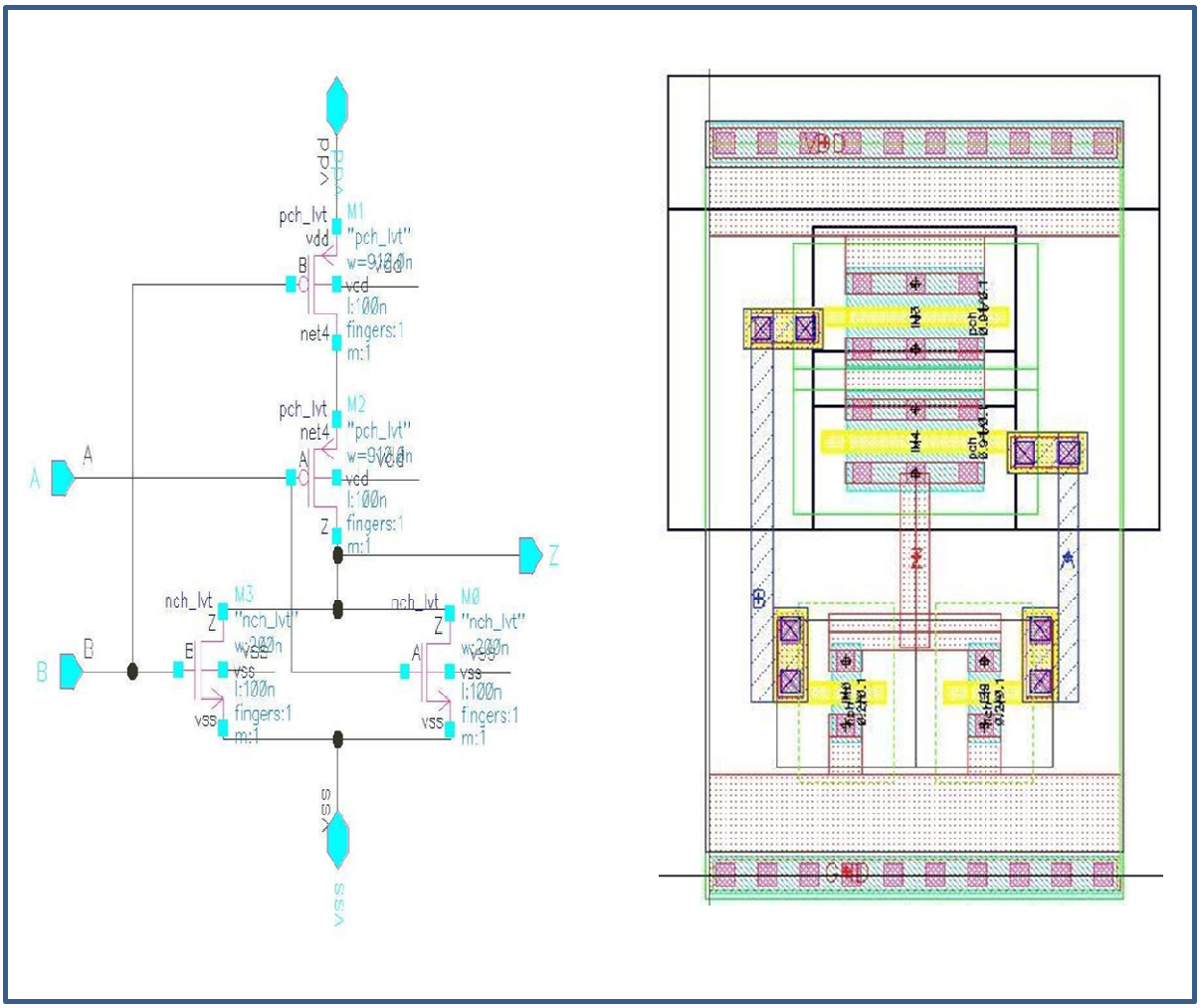


Figure 37: 90nm NOR schematic and layout

Table 9: NOR delay characteristics at 45nm

	Rise Time	Fall Time	Delay	Total Power
Worst Case	9.06ps	8.085ps	5.60ps	1.19uW
Nominal	8.81ps	7.16ps	3.94ps	1.42uW
Best Case	8.65ps	7.057ps	3.26ps	1.75uW

3.4 Flip Flop

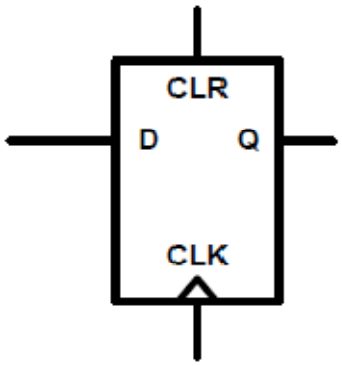


Figure 38: Flip Flop Symbol

Table 10: Flip Flop delay characteristics at 90nm

	Rise Time	Fall Time	Delay	Total Power	setup	hold
Worst Case	64.9ps	73.19ps	137.1ps	22.24uW	38ps	26.9ps
Nominal	47.9ps	52.51ps	99.4ps	51.36uW	31ps	22.8ps
Best Case	38.18ps	40.08ps	80.3ps	79.46uW	23ps	18.36ps

Area = 4.3μm * 16.03 μm = 68.9μm^2

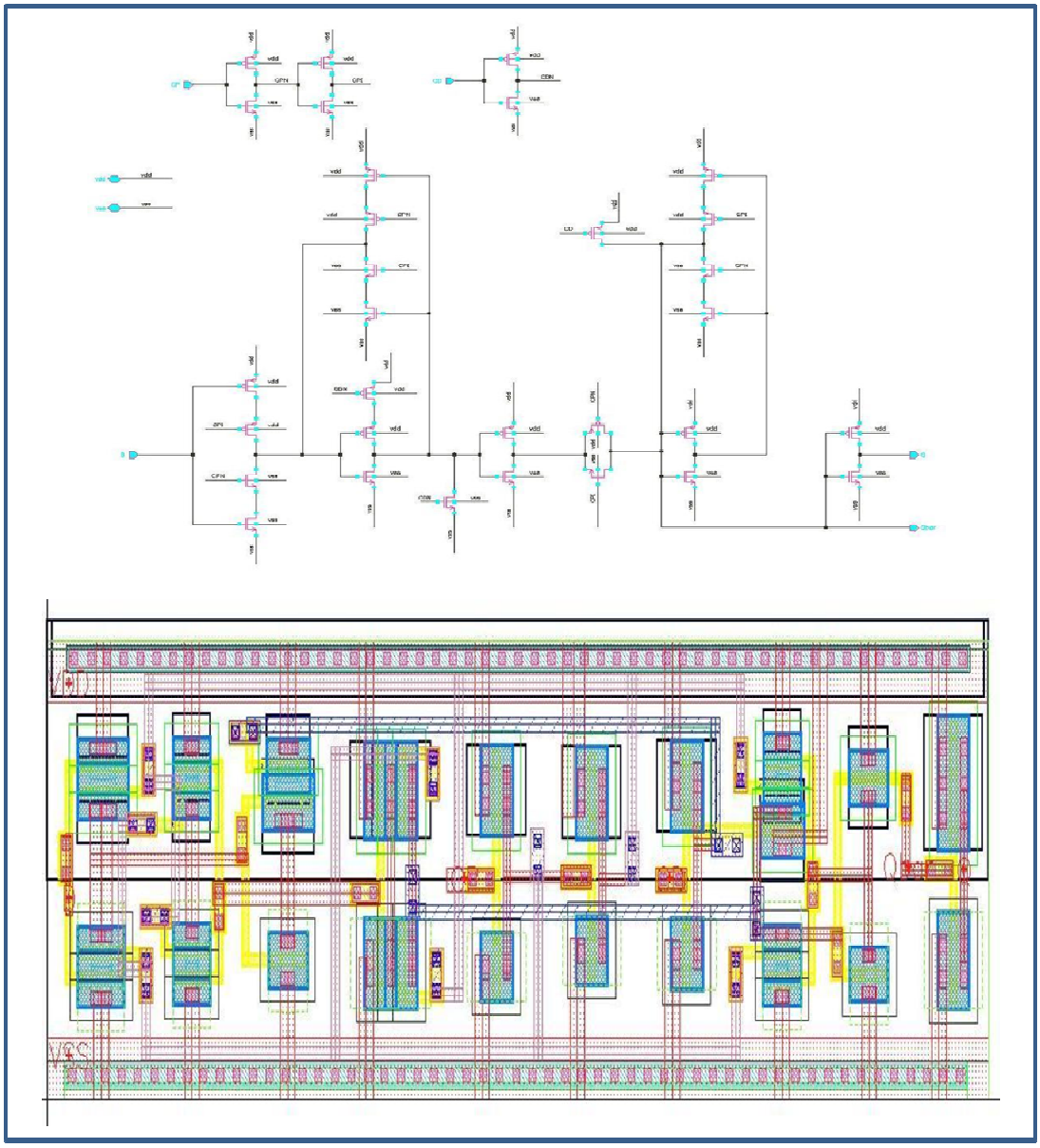


Figure 39: 90nm Flip Flop schematic and layout

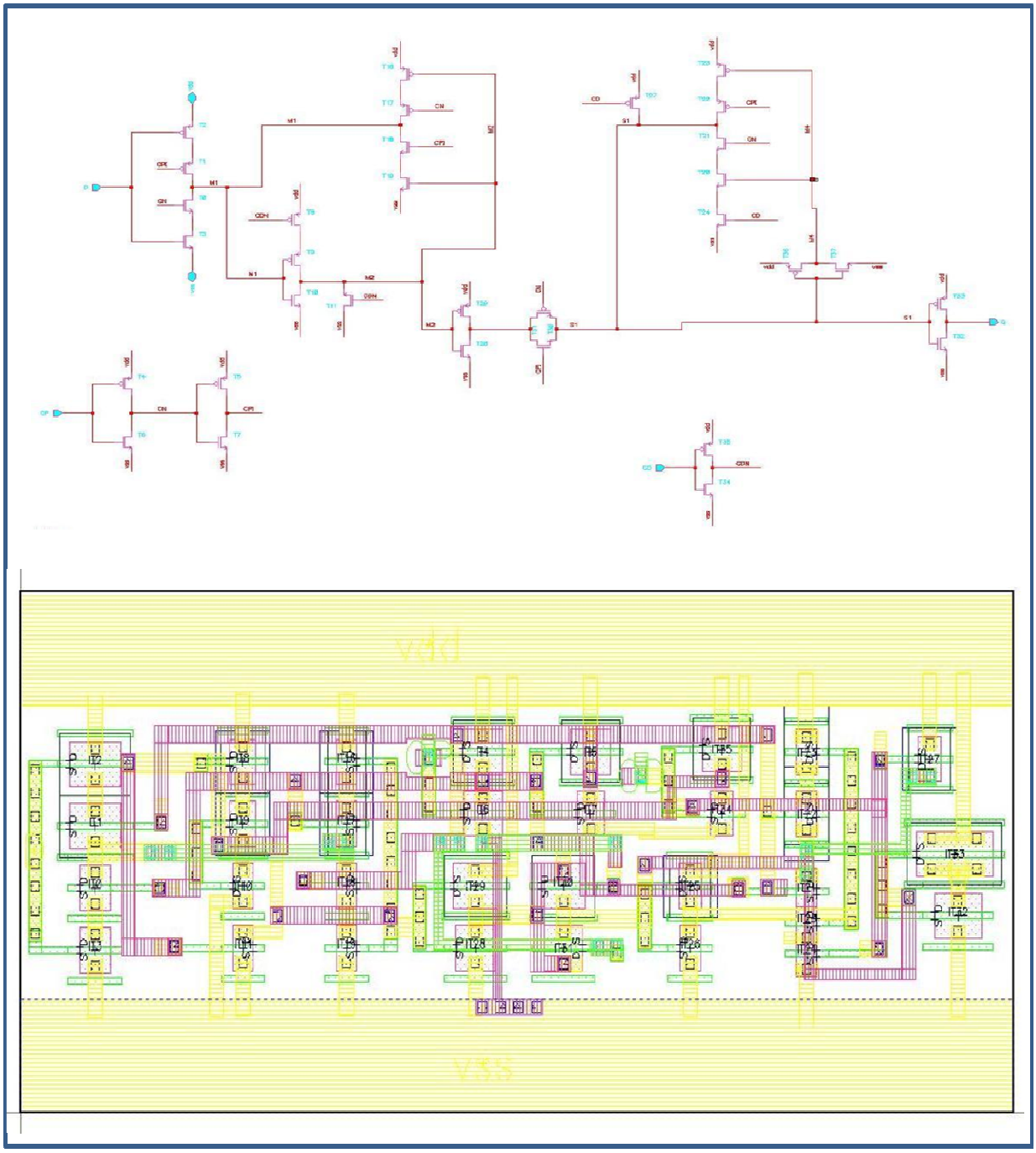


Figure 40: 45nm Flip Flop Schematic and Layout

Table 11: 45nm Flip Flop Delay Characteristics

	Rise Time	Fall Time	Delay	Total Power	setup	hold
Worst Case	6.33ps	4.807ps	28.11ps	6.83uW	10ps	4.97ps
Nominal	5.01ps	4.11ps	19.82ps	8.41uW	9ps	4.09ps
Best Case	4.91ps	3.94ps	17.80ps	8.73uW	8ps	3.83ps

Area = 3.2μm * 7.648 μm = 24.47μm^2

3.5 XOR

Figure 41: XOR Symbol

Table 12: XOR Truth Table

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Table 13: 90nm XOR Delay characteristics

	Rise Time	Fall Time	Delay	Total Power
Worst Case	35.77ps	25.06ps	48.06ps	10.818uW
Nominal	26.82ps	22.74ps	39.66ps	11.91uW
Best Case	22.17ps	18.34ps	34.11ps	13.89uW

Area = 4.3μm * 3.86 μm = 16.59μm^2

Table 14: 45nm XOR delay characteristics

	Rise Time	Fall Time	Delay	Total Power
Worst Case	13.12ps	7.765ps	6.76ps	3.14uW
Nominal	11.57ps	6.47ps	5.97ps	3.6uW
Best Case	10.55ps	5.85ps	5.46ps	4.13uW

Area = 3.2μm * 5.377 μm = 17.20μm^2

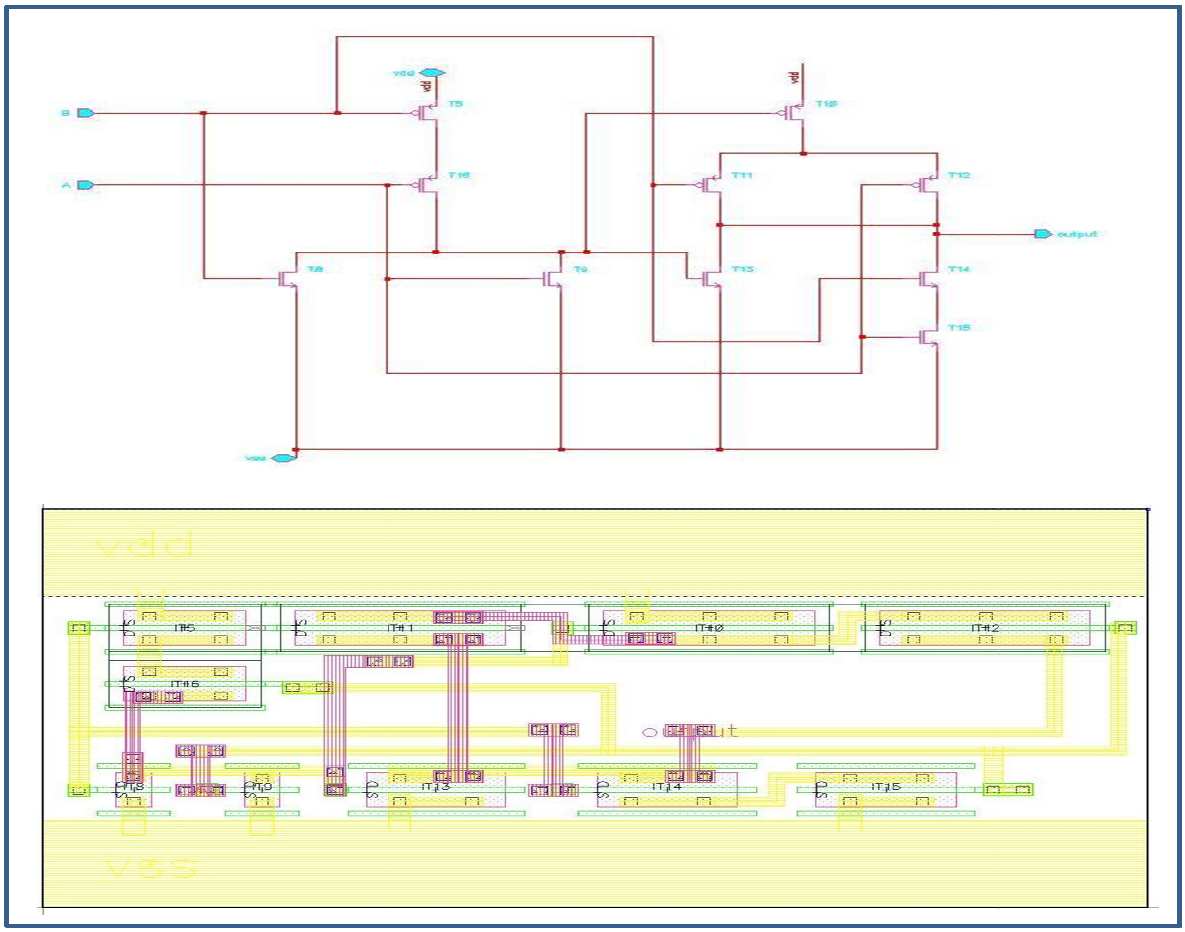


Figure 42: 45nm XOR schematic and layout

The standard cell library is used to implement more complex logic. Some of these logic functions are as follows.

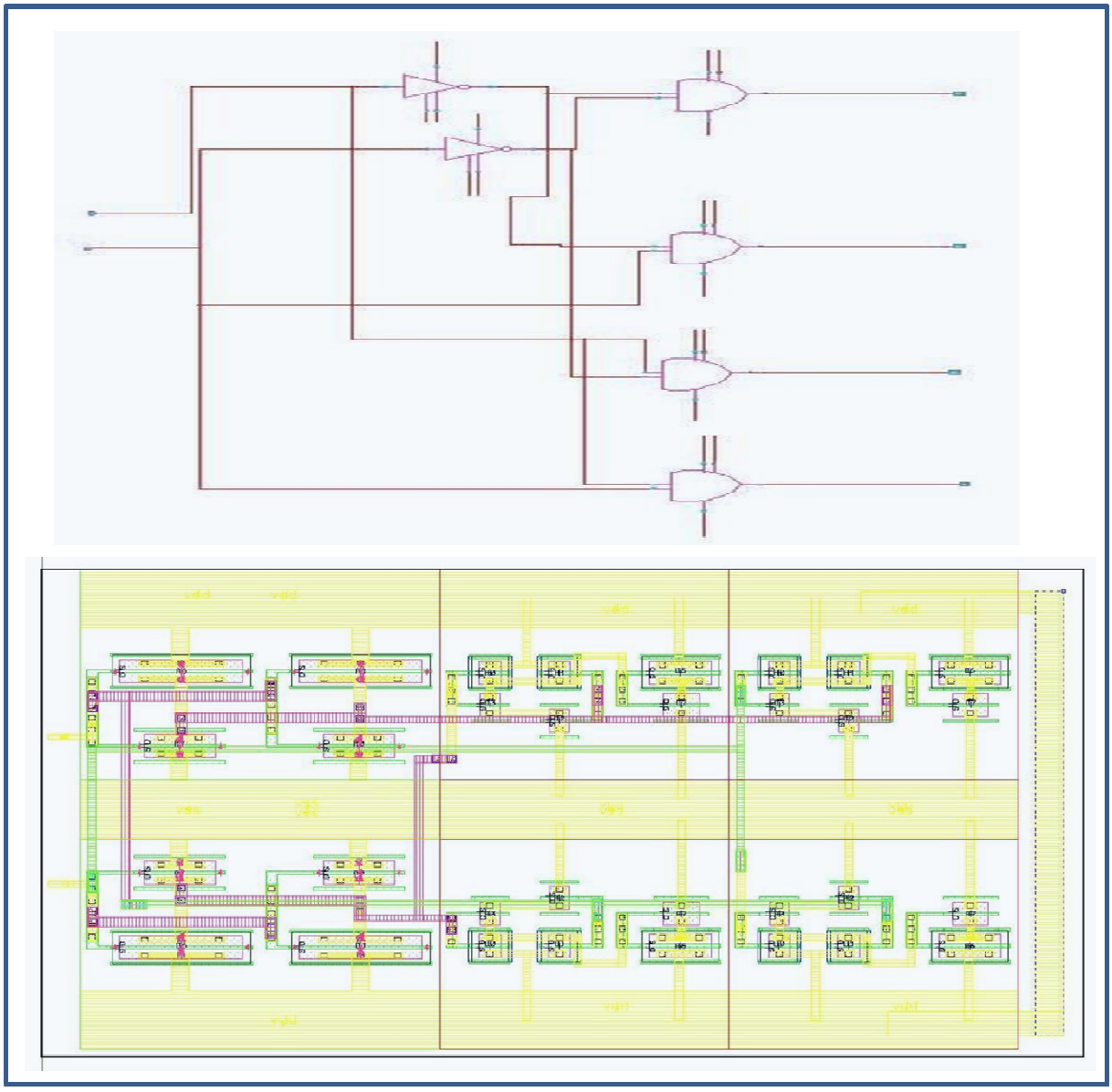


Figure 43: 45nm 2:4 Decoder schematic and layout

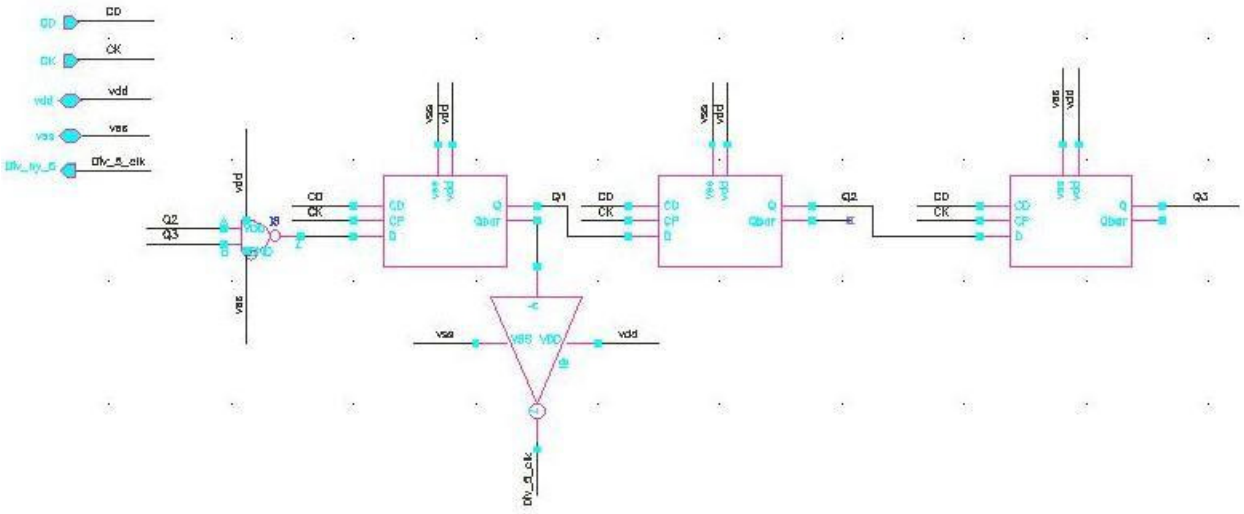


Figure 44: 90nm Ripple Clock divider (/5) schematic

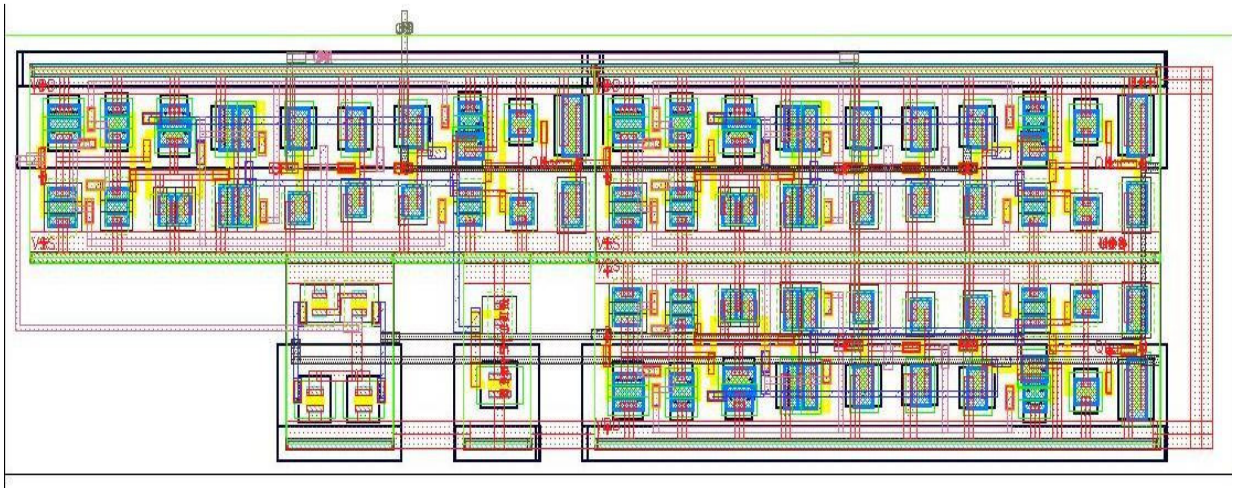


Figure 45: 90nm Ripple Clock divider (/5) layout

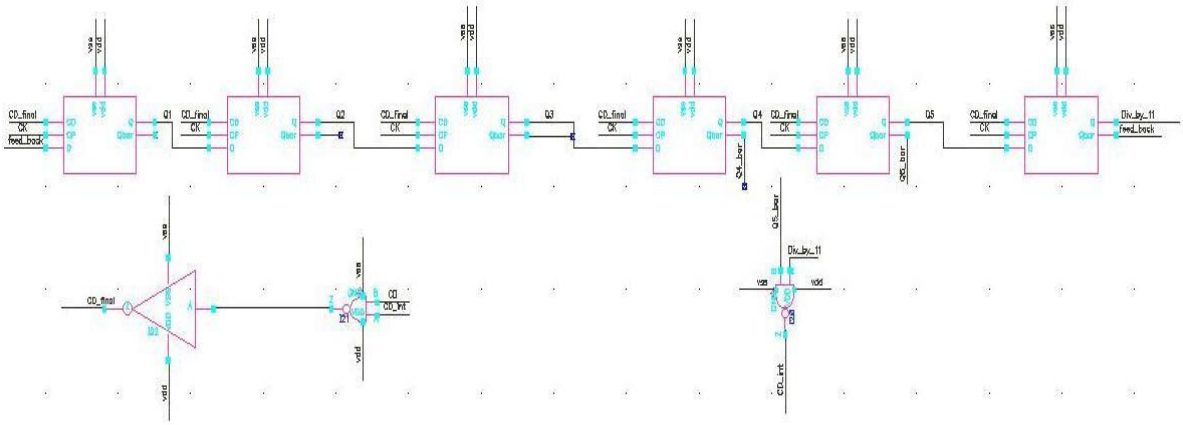


Figure 46: 90nm Ripple Clock divider (/11) schematic

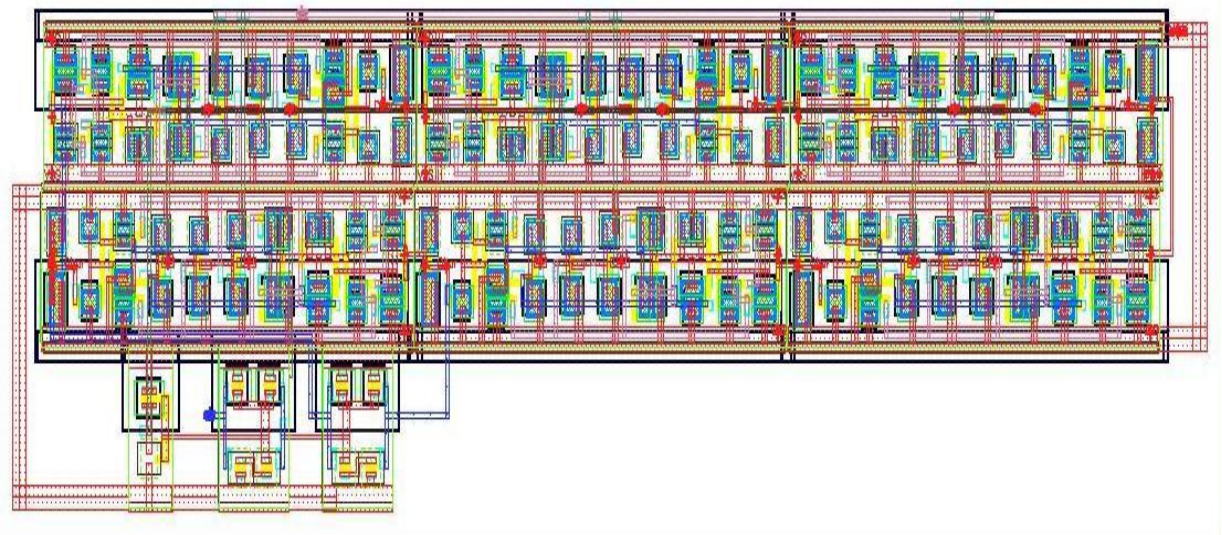


Figure 47: 90nm Ripple Clock divider (/11) layout

4 NON-COHERENT AND COHERENT DEMODULATION

4.1 Need for Modulation

A wide variety of signals are encountered in communication systems, and many of these signals are not suited for direct transmission in the case of atmosphere being used as a medium of transmission. As a result of the wide variety of signals, there exists a wide band of frequencies that cannot be propagated. Hence a transformation is required to translate this signal information into higher frequencies thereby reducing the percentage signal bandwidth thereby making it suitable for transmission. This transformation is known as modulation. Some of the benefits of modulation are as follows [25]

Transmission Efficiency: Since the signals are translated to higher frequencies it makes amplifier and antennae systems easier to design i.e. size of the antennae will be more manageable and also the use of different carrier frequencies during modulation help in the isolation of signals transmitted from multiple similar transmitters.

Reduced Bandwidth: Bandwidth of the signal is reduced due to modulation which in turn reduces the signal to noise ratio which is a function of signal bandwidth.

There are several modulation schemes, and the process of choosing from among these is typically based on certain performance parameters such as signal to noise ratio, bandwidth efficiency etc. The various modulation schemes can be broadly classified as Analog

modulation techniques and Digital modulation techniques. Analog modulation techniques are Amplitude modulation (AM), Frequency modulation (FM) and Phase modulation (PM) and the corresponding Digital modulation techniques are Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase shift Keying (PSK).

Communication systems are generally bounded by constraints such as

1. Available Bandwidth.
2. Permissible Power.
3. Inherent noise of the system.

Digital modulation techniques have more information capacity, are compatible with digital data services, offer better data security, offer better quality communications and offer quicker system availability when compared with Analog modulation techniques [25].

Once the appropriate Digital modulation scheme is implemented at Transmitter side, there should also be a corresponding demodulator at the receiver. Digital demodulation can be achieved coherently or non-coherently.

Coherent demodulation requires a reference carrier signal at the demodulator in order to achieve error free or minimum error data recovery from the modulated data. In most coherent demodulation schemes the incoming modulated data is a 90 degree signal and the in-phase and quadrature phase components are recovered from the incoming signal. However in most cases the carrier reference is not available at the receiver and must be recovered from the incoming signal. This carrier recovery process is complex and forms a

significant part of the coherent demodulator. As a result the hardware required to implement a coherent demodulator is significant [34].

In real communication environments signals are affected by variations in amplitude and phase due to frequency selective fading channel or phase noise induced due to VCO. In cases where carrier recovery circuit is difficult to implement especially for high frequency applications such as millimeter wave applications where the time delay associated with the synchronizing circuitry may cause degradation in the throughput of the system, thereby making Non coherent demodulation a attractive option [36]. When the information is differentially encoded, Non coherent demodulation uses the phase difference between two successive data sent to identify the data.

4.2 Non Coherent Demodulation

The complex multiplier is at the heart of the Non Coherent Demodulator.

4.2.1 Complex Multiplier

The complex multiplier calculates the cross and dot product of the current and previous data, which is denoted in complex form generated at the output of the two receiver matched filters.

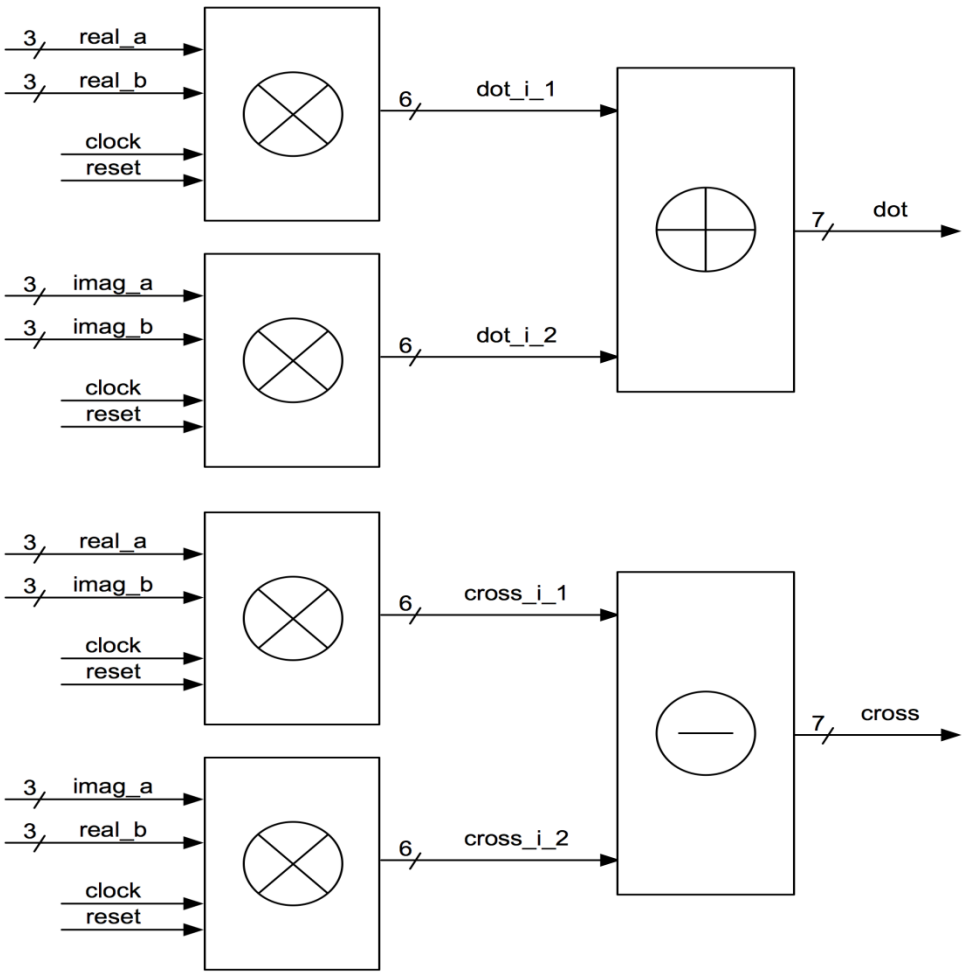


Figure 48: Block Diagram of Complex Multiplier

Let S_k and S_{k-1} represent the current complex received symbol and previous complex received symbol, respectively. With $S_k = I_k + jQ_k$

The complex multiplication between these two symbols becomes:

$$S_k S_{k-1}^* = (I_k + jQ_k)(I_{k-1} - jQ_{k-1}) = (I_k I_{k-1} + Q_k Q_{k-1}) + j(Q_k I_{k-1} - I_k Q_{k-1})$$

The Dot and Cross products can then be defined as:

$$\mathrm{Dot(k)} = \mathrm{I_k I_{k-1}} + \mathrm{Q_k Q_{k-1}}$$

$$\mathrm{Cross(k)} = \mathrm{Q_k I_{k-1}} - \mathrm{I_k Q_{k-1}}$$

Examination of these products in the complex plane reveals that the Dot and Cross products are the real and imaginary results of complex multiplication of the current and previous symbols. The Dot product alone thus allows determination of the phase shift between successive BPSK symbols, while the Dot and Cross products together allow determination of the integer number phase shifts between successive QPSK symbols. Differential encoding of the source data implies that an absolute phase reference is not required, and thus knowledge of the phase shift between successive symbols derived from the Dot and Cross products unambiguously permits correct demodulation.

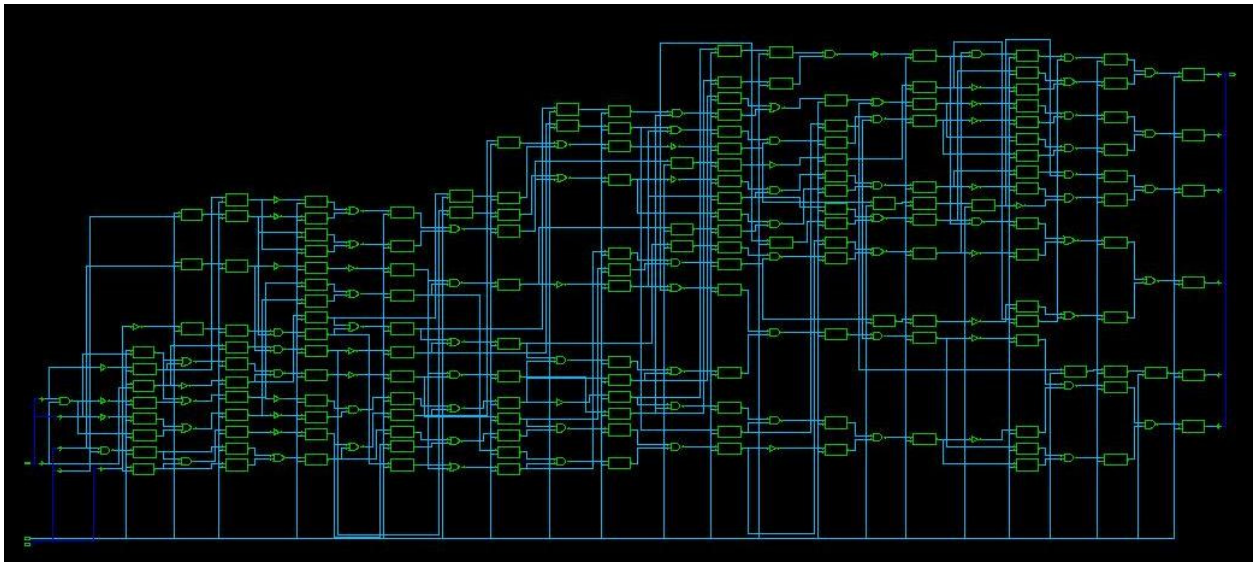


Figure 49: Schematic of Complex Multiplier

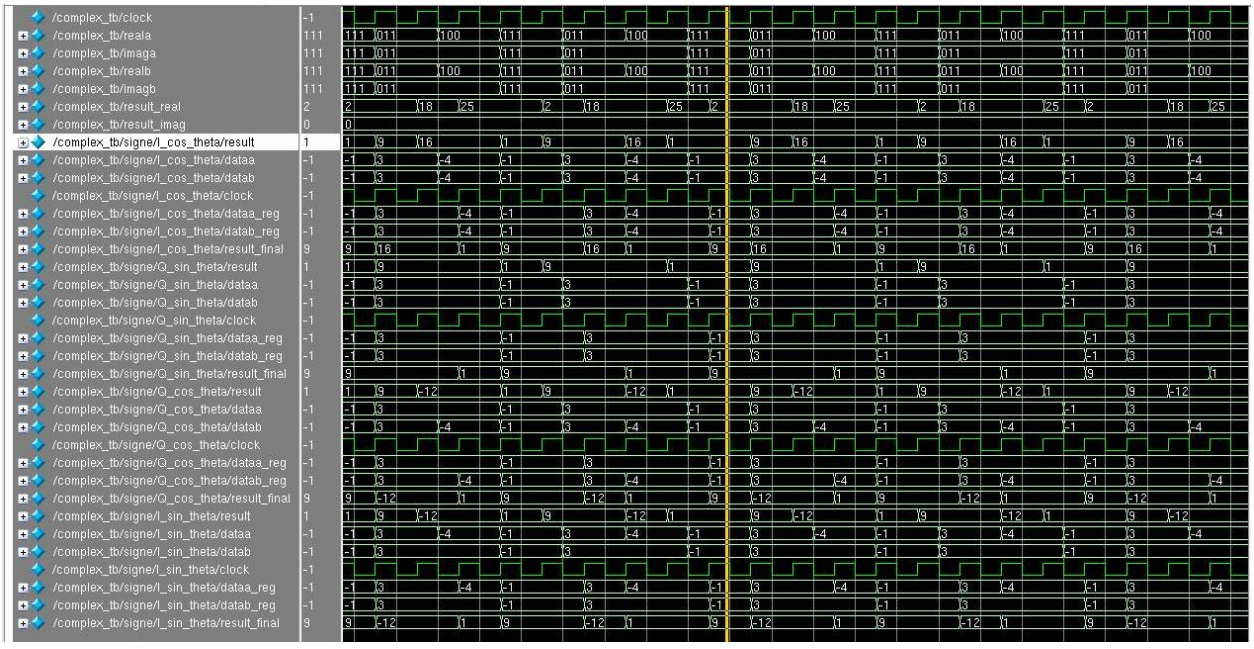


Figure 50: Simulation results of Complex multiplier

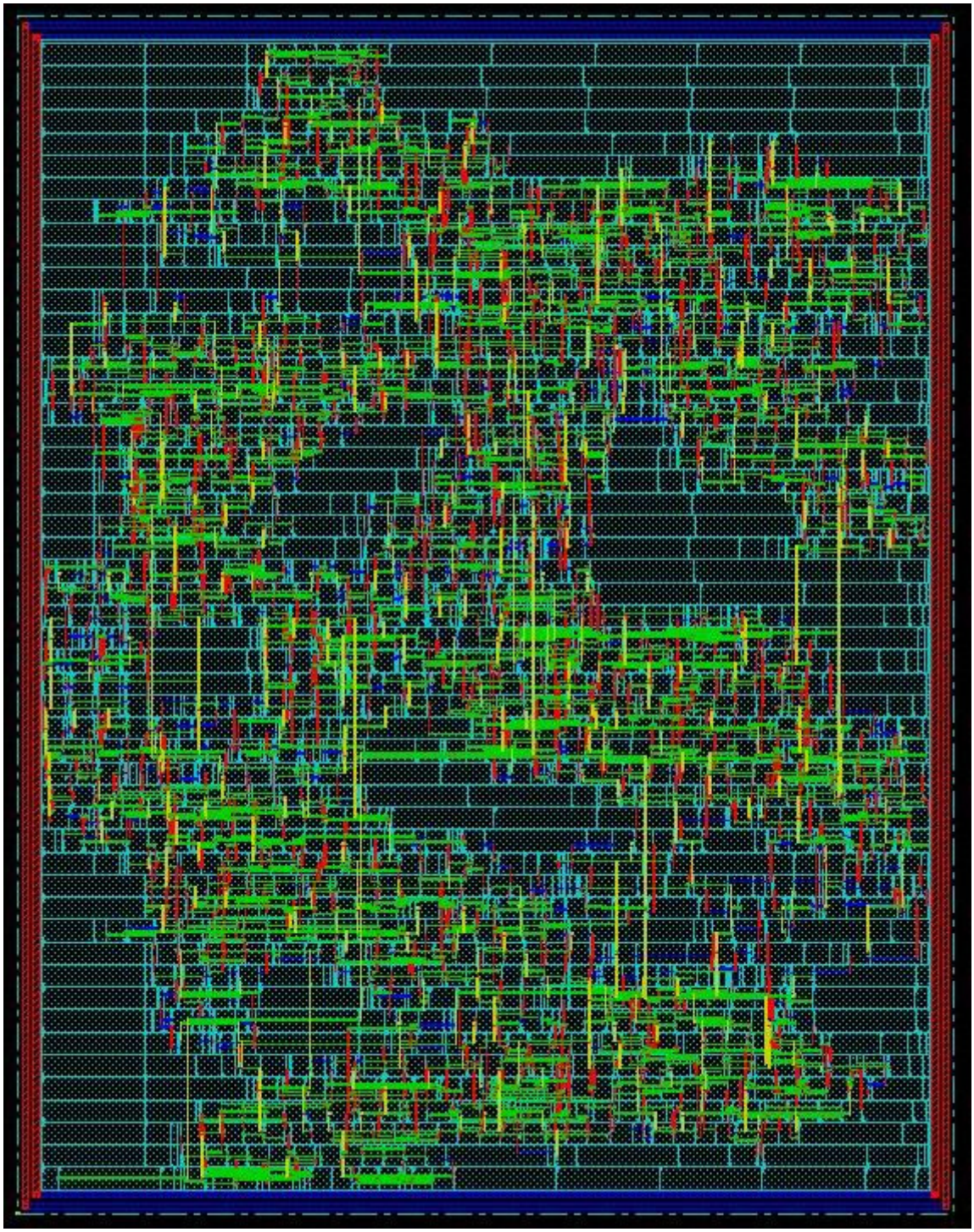


Figure 51: Layout of Complex multiplier

Table 15: 90nm Complex Multiplier design information

Block name	Gate count	Flip-Flop area	Combinational area	Max.freq. (GHz)
3-bit multiplier	535	2561	375	3.220
6-bit adder	753	3726	430	3.300
6-bit subtractor	735	3561	472	3.220
3-bit complex multiplier	3209	15390	2219	3.125

4.3 Coherent Demodulation

Coherent demodulation is carried out using the Costas Loop [35][26] . The Digital Costas loop performs the baseband task required for the demodulation of waveforms such as BPSK, QPSK. Some of these tasks are carrier tracking and symbol synchronization.

4.3.1 Costas Loop – Principle of Operation

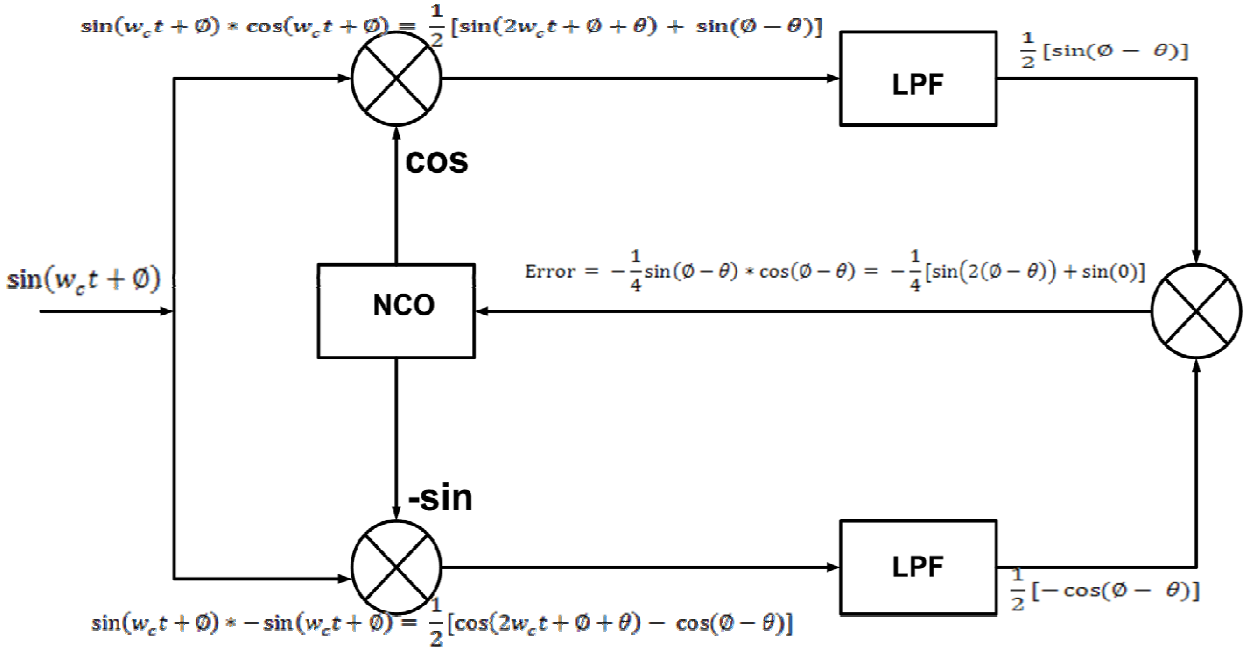


Figure 52: Block diagram of Costas Loop

Assume that the incoming signal is a single tone with an unknown frequency (w_c) and phase ($\emptyset$). Assume that the frequency at the receiver is exactly matched with the incoming signal. The incoming signal is given by

$$\sin(w_c t + \emptyset)$$

The input is mixed with the cosine and sine wave generated by the NCO. The system here is performing down conversion i.e.

$$\sin(w_c t + \emptyset) * \cos(w_c t + \emptyset) = \frac{1}{2}[\sin(2w_c t + \emptyset + \theta) + \sin(\emptyset - \theta)]$$

$$\sin(w_c t + \emptyset) * -\sin(w_c t + \emptyset) = \frac{1}{2}[\cos(2w_c t + \emptyset + \theta) - \cos(\emptyset - \theta)]$$

This mixing results in sum and difference term with respect to the phase. The Low pass filter following the mixer is designed to eliminate the sum signal out, hence allowing only the difference term to go through. The signal at the output of the Low Pass filter is as follows:

$$\frac{1}{2}[\sin(\emptyset - \theta)]$$

$$\frac{1}{2}[-\cos(\emptyset - \theta)]$$

The signal outputs of the Low pass filters on both arms are mixed with each other resulting in a sum and a difference term i.e.

$$\text{Error} = -\frac{1}{4}\sin(\emptyset - \theta) * \cos(\emptyset - \theta) = -\frac{1}{4}[\sin\big(2(\emptyset - \theta)\big) + \sin(0)]$$

Here the difference term is zero; the resulting term is the sum term which is a function of the difference in the phase. This forms the error signal. It is important to note the negative sign in the error signal indicating that it pushes the loop in the direction opposite to the error thereby helping the loop to lock. It is preferred that the error signal have a linear relationship with the difference in phase, therefore for small values of θ,

$$\sin\theta \cong \theta$$

Hence it can be assumed that the error signal is linearly dependent on the phase difference$(\emptyset - \theta)$.

The block diagram of Costas loop for Baseband is as follows:

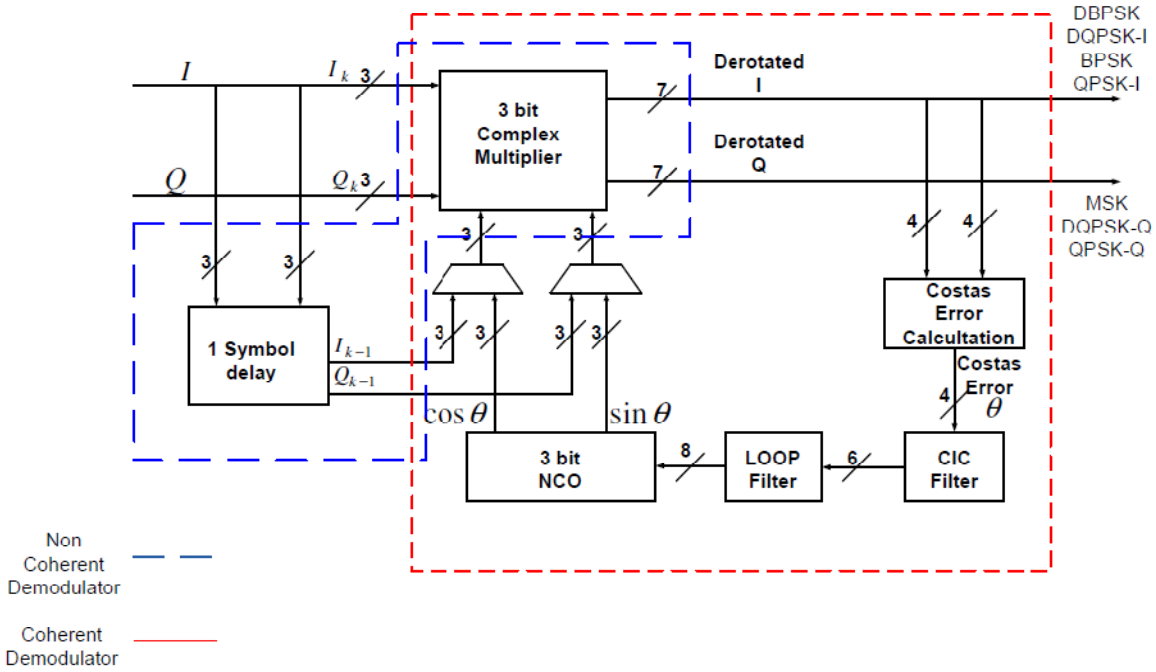


Figure 53: Baseband Costas Loop

4.3.2 Digital Derotator

The derotation is done by the complex multiplier with cosine and sine inputs from the numerically controlled oscillator (NCO). The derotation process can be represented as follows.

The input data streams are I and Q

$$I = \cos\theta$$

$$Q = \sin\theta$$

Inputs from the NCO

$$I = \cos\theta_1$$

$$Q = \sin\theta_1$$

The derotated data is as follows:

$$I_{rotated} = \cos(\theta - \theta_1) = \cos\theta\cos\theta_1 + \sin\theta\sin\theta_1 = I\cos\theta_1 + Q\sin\theta_1$$

$$Q_{rotated} = \sin(\theta - \theta_1) = \sin\theta\cos\theta_1 - \cos\theta\sin\theta_1 = Q\cos\theta_1 - Q\sin\theta_1$$

4.3.3 Costas Error Calculation

The error between the derotated I and Q signals for BPSK and QPSK is calculated as follows:

$$\theta_{BPSK} = Q\,\mathrm{sign}(I)$$

$$\theta_{QPSK} = Q\,\mathrm{sign}(I) - I\,\mathrm{sign}(Q)$$

This error signal is forwarded to the CIC decimation filter. This error is decimated and the rest of the loop operates on this error signal at a lower frequency.

4.3.4 CIC Filter

CIC filters [27] [37] are used in multi rate processing; Implementation of CIC filters are advantageous when compared to FIR filters because CIC filters only use addition and subtraction, while most FIR filters use multiplication. Moreover CIC filters have specific frequency roll-off while low pass FIR filters do not have a single frequency roll-off. CIC filters can be used for either decimation or interpolation. The CIC filter in the costas loop is used for the purpose of decimation. The CIC filter as a decimator is used in the costas loop since the loop bandwidth is small compared to the sampling frequency. In the implementation here a decimation of 10 is used i.e. the CIC filter discards 9 out of every 10 samples i.e. decimating the frequency from 2.5Ghz to 250 Mhz.

The basic building blocks of CIC filter are an Integrator and a Comb filter. Typically a CIC filter consists of a cascade of comb filters connected to a cascade of integrator stages.

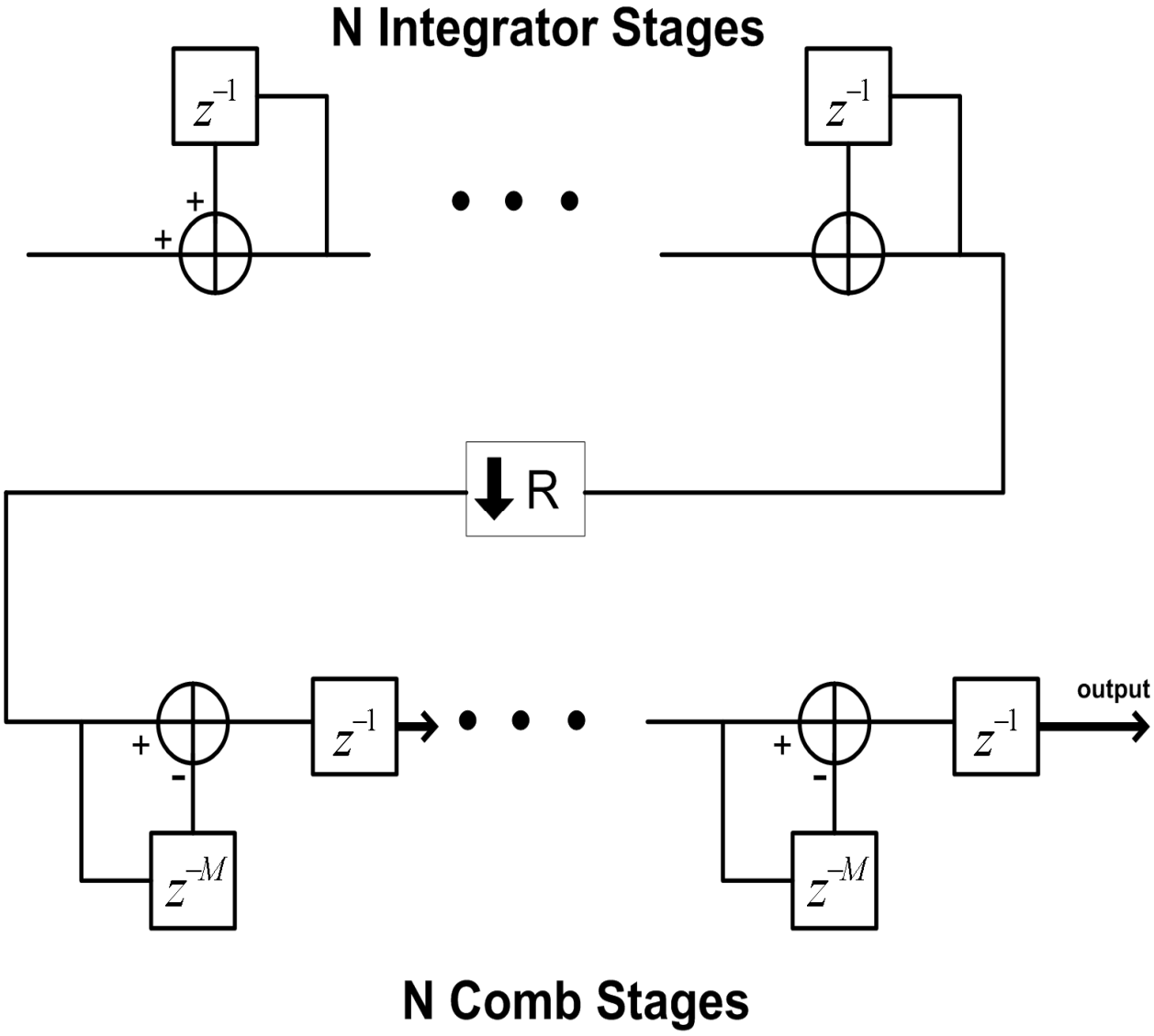


Figure 54: Block diagram of CIC filter

4.3.4.1 Integrator

An integrator (accumulator) essentially is a single pole (pole at 0) IIR filter with unity feedback coefficient. Integrator can be described as:

$$y[n] = y[n-1] + x[n]$$

The transfer function for an integrator is given as:

$$H[z] = \frac{1}{1 - z^{-1}}$$

The output of an integrator can grow unbounded for a bounded input i.e. an Integrator by itself is unstable.

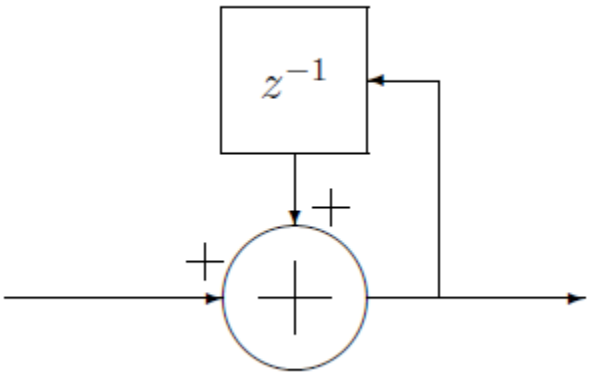


Figure 55: Integrator

When the CIC filter is used for the purpose of decimation, although the comb filter functions at a lower frequency i.e. decimated frequency, the Integrator on the other hand still needs to work at full frequency of the input. Therefore the Integrator/accumulator needs to be pipelined to achieve high operating speeds. One such architecture for a pipelined accumulator/Integrator [28]is given below.

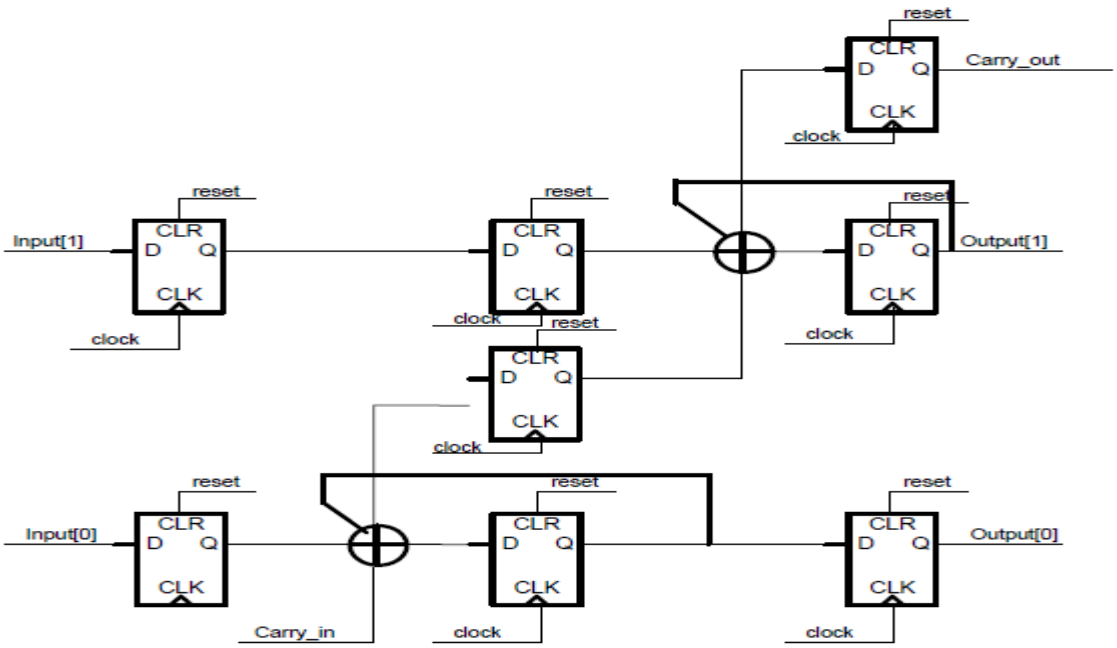


Figure 56: Pipelined Accumulator

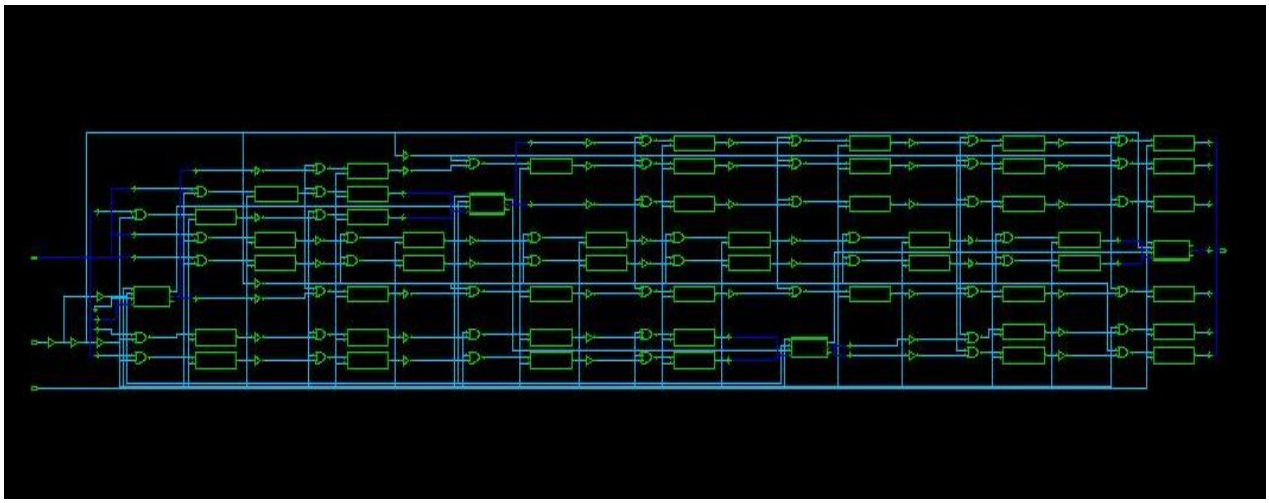


Figure 57: Schematic of integrator

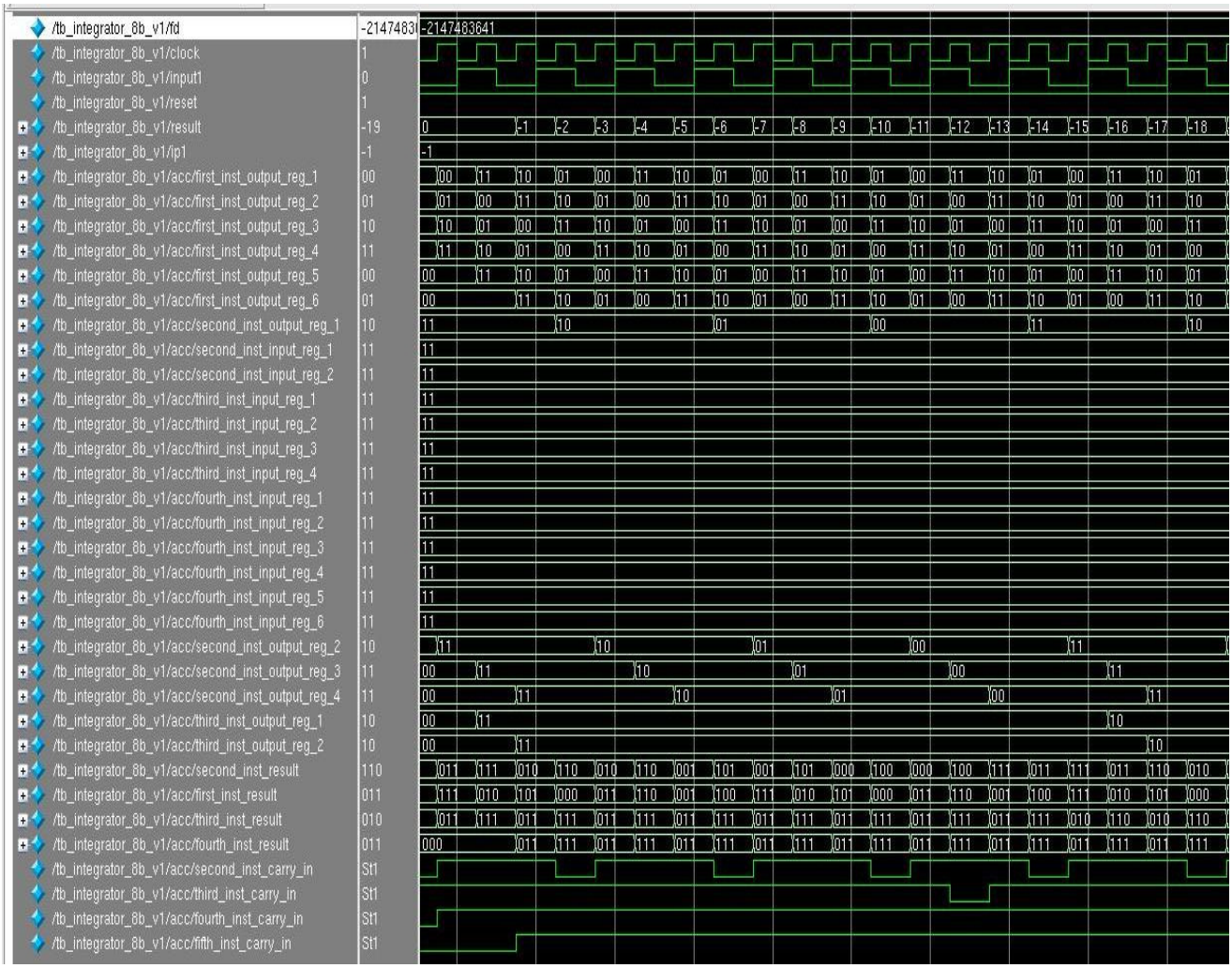


Figure 58: Simulation result of Integrator

4.3.4.2 Comb Filter

Comb filter [29] is called so because the frequency response of a comb filter consists of regularly spaced spikes which look similar to a comb.

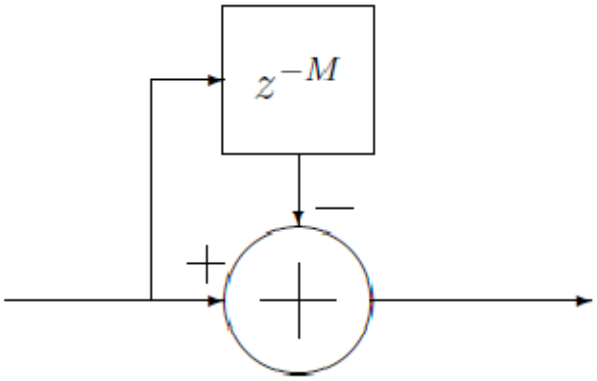


Figure 59: Comb filter

Comb filter can expressed as follows:

$$y[x] = x[n] + x[n - M]$$

The transfer function of a comb filter is given by:

$$H[z] = 1 + z^{-M}$$

The transfer function for a CIC filter with a decimation ratio of R and a differential delay parameter of M with N stages of comb filter and integrator each is given by

$$H[z] = \frac{(1 - z^{RM})^N}{(1 - z^{-1})^N}$$

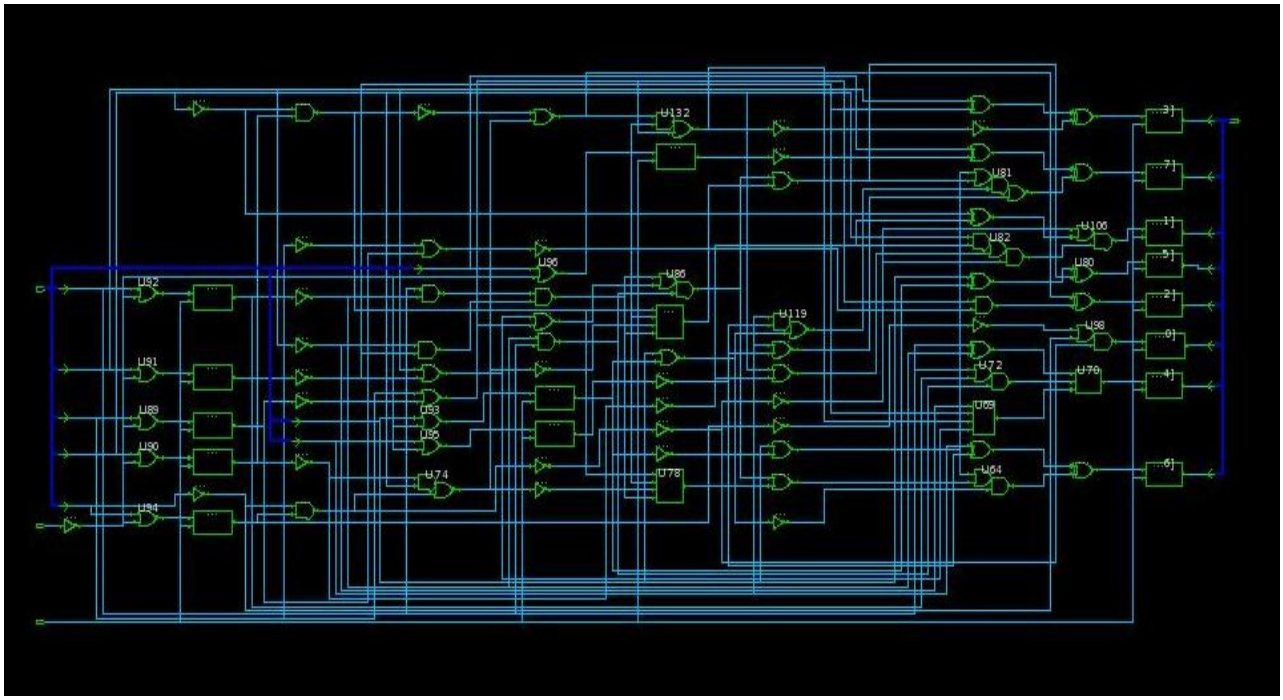


Figure 60: Schematic of Comb Filter

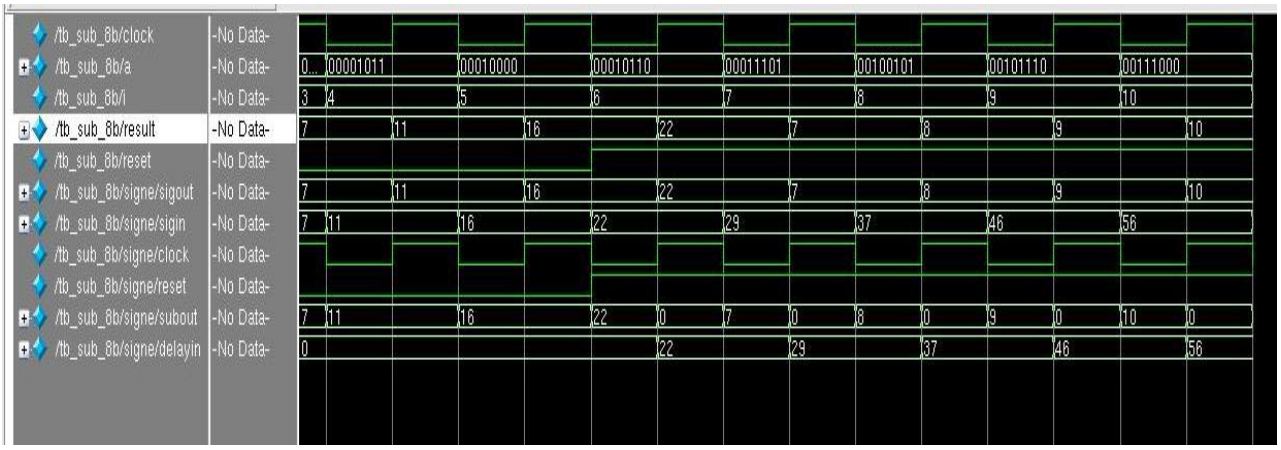


Figure 61: Simulation results for Comb Filter

4.3.4.3 Bit Growth

The Gain of the CIC filter with a decimation ratio of R and a differential delay parameter of M with N stages of comb filter and integrator each is given by:

$$\text{Gain} = (\text{R} * \text{M})^{\text{N}}$$

If B_{in} is the number of bits at the input, then the number of bits at the output i.e. the growth of bits is given by:

$$\text{B}_{\text{out}} = [\text{N} * \log_2(\text{R} * \text{M}) + \text{B}_{\text{in}}]$$

The above equation indicates that B_{out} bits are required at each of the integrator and comb filter stages, therefore the input is sign extended to B_{out} bits and then applied to the CIC filter.

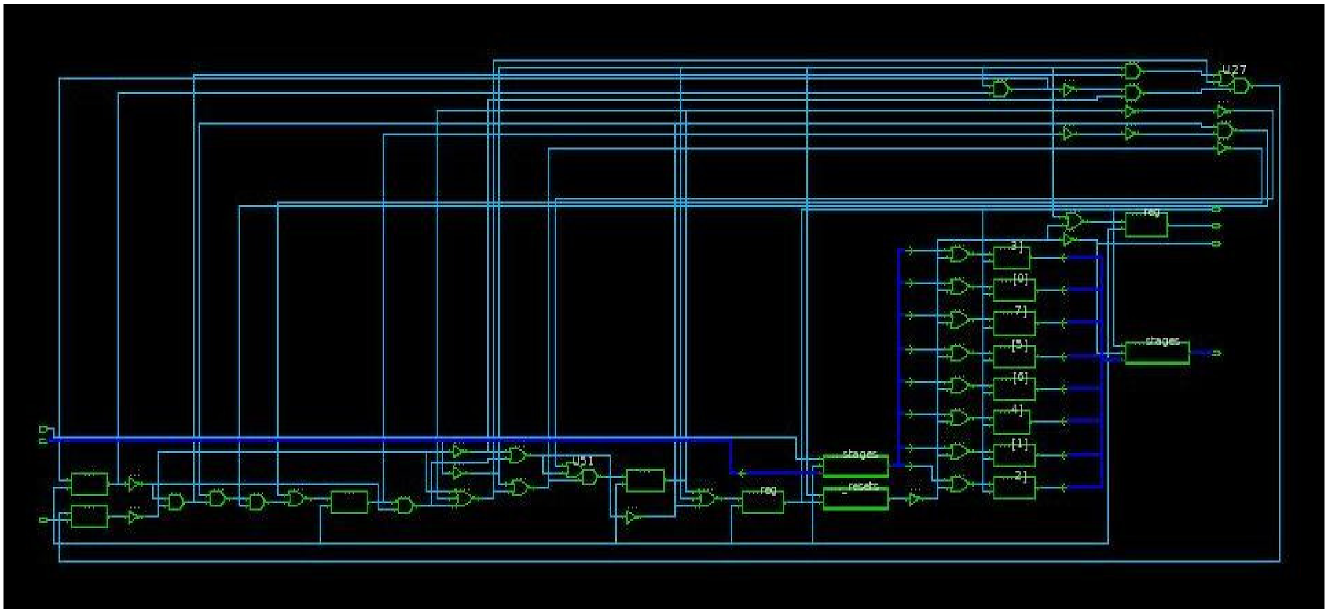


Figure 62: Schematic of CIC filter

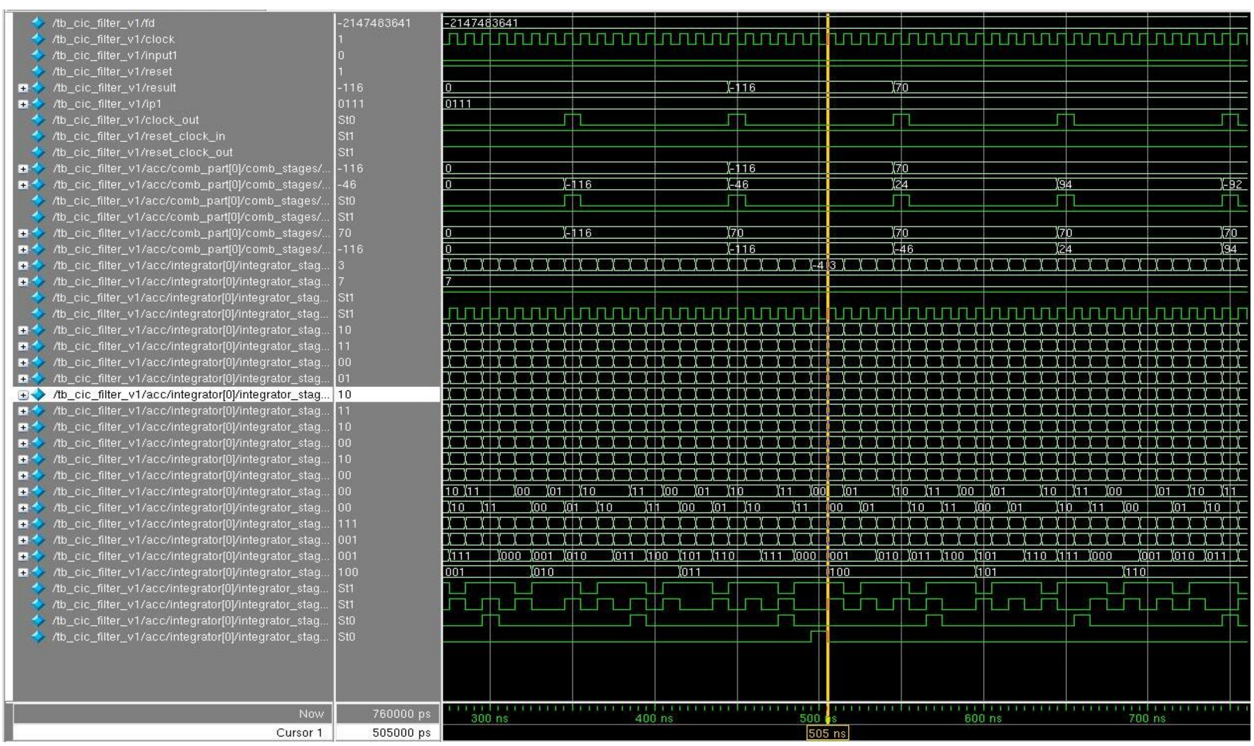


Figure 63: Simulation results for CIC filter

Figure 64: Layout of CIC filter

Table 16: 90nm CIC filter design information

Block name	Flip Flop Area	Combinational Area	Max Freq	Average Power
reset synch	37	0	N/A	0.01mW
comb filter	230	198	250Mhz	0.175mW
integrator	1281	1270	3.1Ghz	6mW
cic_filter	1770	1740	N/A	7.175mW

4.3.5 Loop Filter

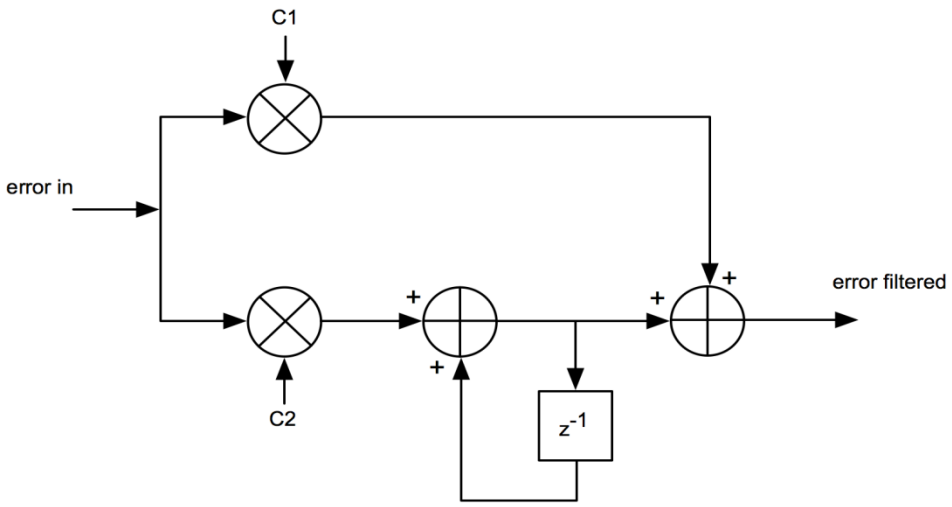


Figure 65: Block diagram of Loop Filter

The loop filter [30] illustrated above is a first order loop filter for a second order loop. The order can be obtained from the transfer function i.e. the highest power of z in the denominator indicates the order of the loop.

The phase error (error_in) is multiplied by proportional gain constant c1 (proportional gain constant) in the upper arm. In the lower arm the signal error_in is first multiplied by c2 (integral gain constant). The result of this multiplication is fed into an integrator. Finally the sum of the product of proportional gain constant c1 and the phase error (error_in) and the output of the integrator is calculated.

The transfer function for the loop filter can be derived as follows:

$$y = c_1x + y_1$$

$$y_1 = y - c_1x$$

$$y_1 = y_2z^{-1}$$

$$y_2 = c_2x + y_1$$

$$y_1 = (c_2x + y_1)z^{-1}$$

$$y_1 = (c_2x + y - c_1x)z^{-1}$$

$$y = c_1x + (c_2x + y - c_1x)z^{-1}$$

$$y = c_1x + c_2xz^{-1} + yz^{-1} - c_1xz^{-1}$$

$$y(1 - z^{-1}) = c_1x + c_2xz^{-1} - c_1xz^{-1}$$

$$y(1 - z^{-1}) = x(c_1 + c_2z^{-1} - c_1z^{-1})$$

$$\frac{y}{x} = \frac{(c_1 + c_2 z^{-1} - c_1 z^{-1})}{(1 - z^{-1})}$$

$$\frac{y}{x} = \frac{(c_1 + z^{-1}(c_2 - c_1)}{(1 - z^{-1})}$$

$$\frac{y}{x} = \frac{b_0 + b_1 z^{-1}}{a_0 - a_1 z^{-1}}$$

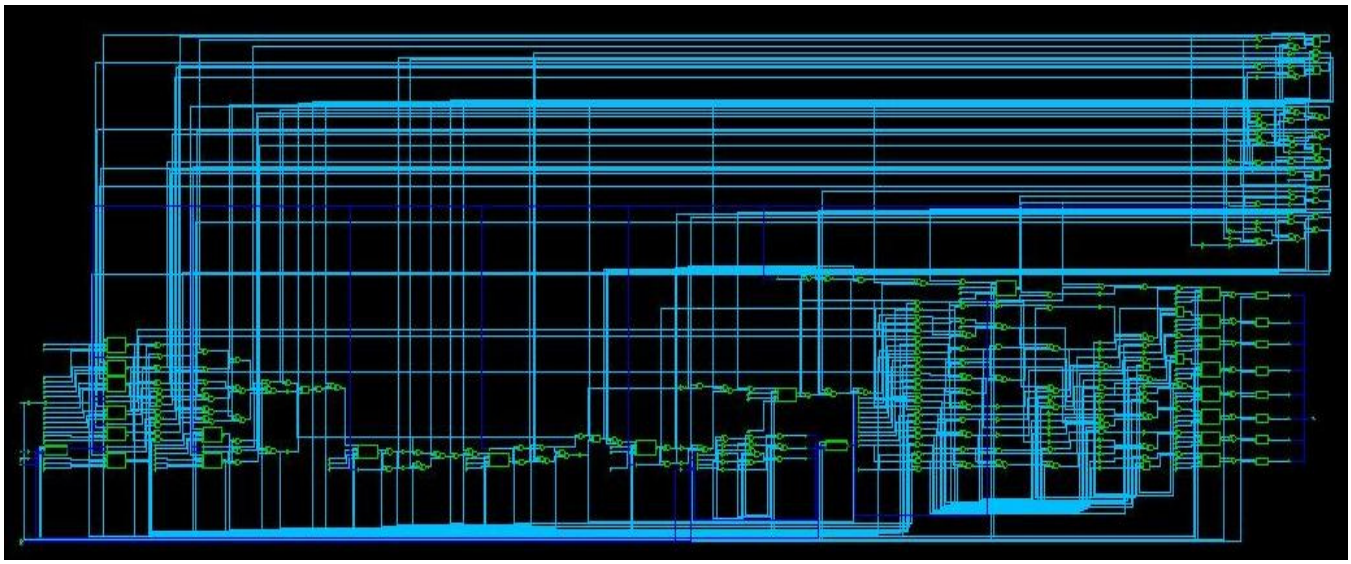


Figure 66: Schematic of Loop Filter

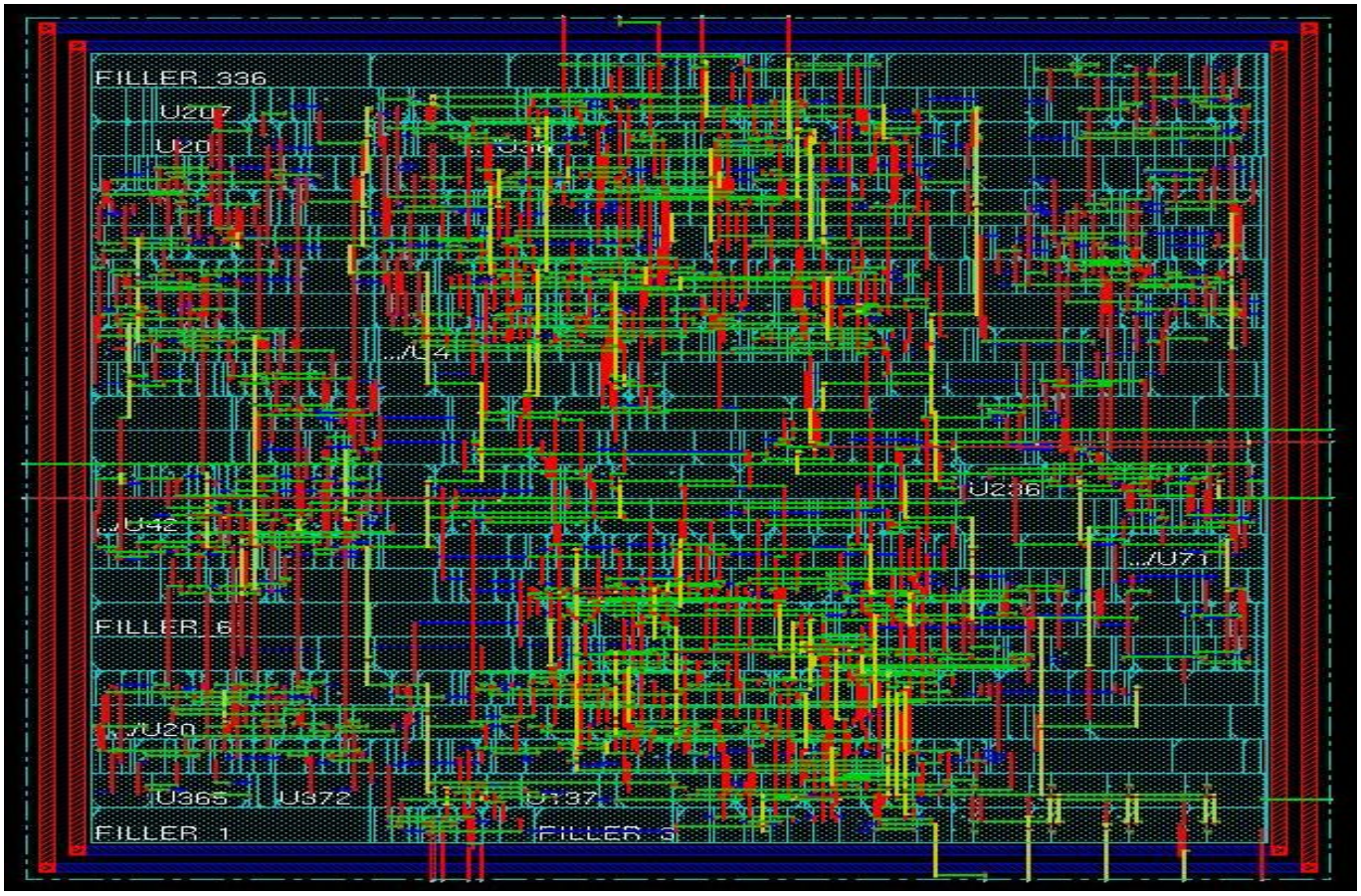


Figure 67: Layout of Loop Filter

Table 17: 90nm Loop filter design information

Block name	Flip Flop Area	Combinational Area	Max Freq	Average Power
8 bit multiplier	571	1876	250	4.3mW
loop filter	685	3417	250	6.7mW

4.3.6 NCO: Numerically Controlled Oscillator

The NCO is used to generate a synchronous discrete arbitrary waveform. The NCO is implemented as a simple look up table. It basically functions as a phase to amplitude

converter. The NCO has a 3 bit output running at a maximum frequency of 250 MHz. The NCO takes in 8 bit input as phase to generate cosine and sine functions, and a maximum of 8 cycles is required to generate a complete cycle. The maximum frequency generated is 15.625 MHz i.e. the loop will lock only for a maximum frequency offset of 15.625 MHz.

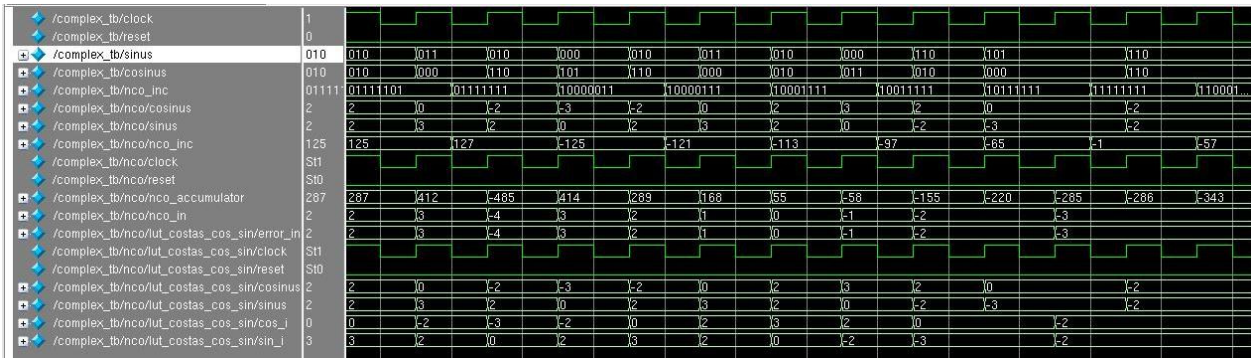


Figure 68: Simulation result of NCO

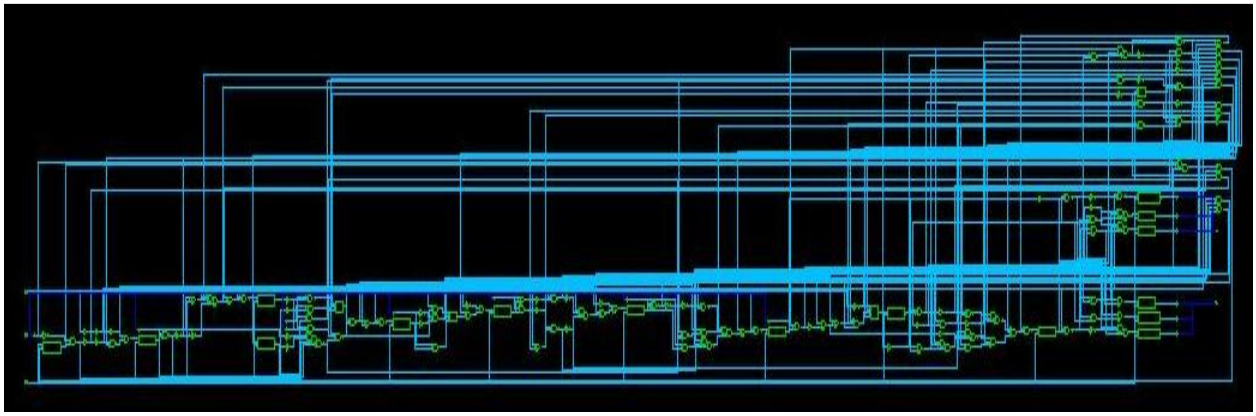


Figure 69: Schematic of NCO

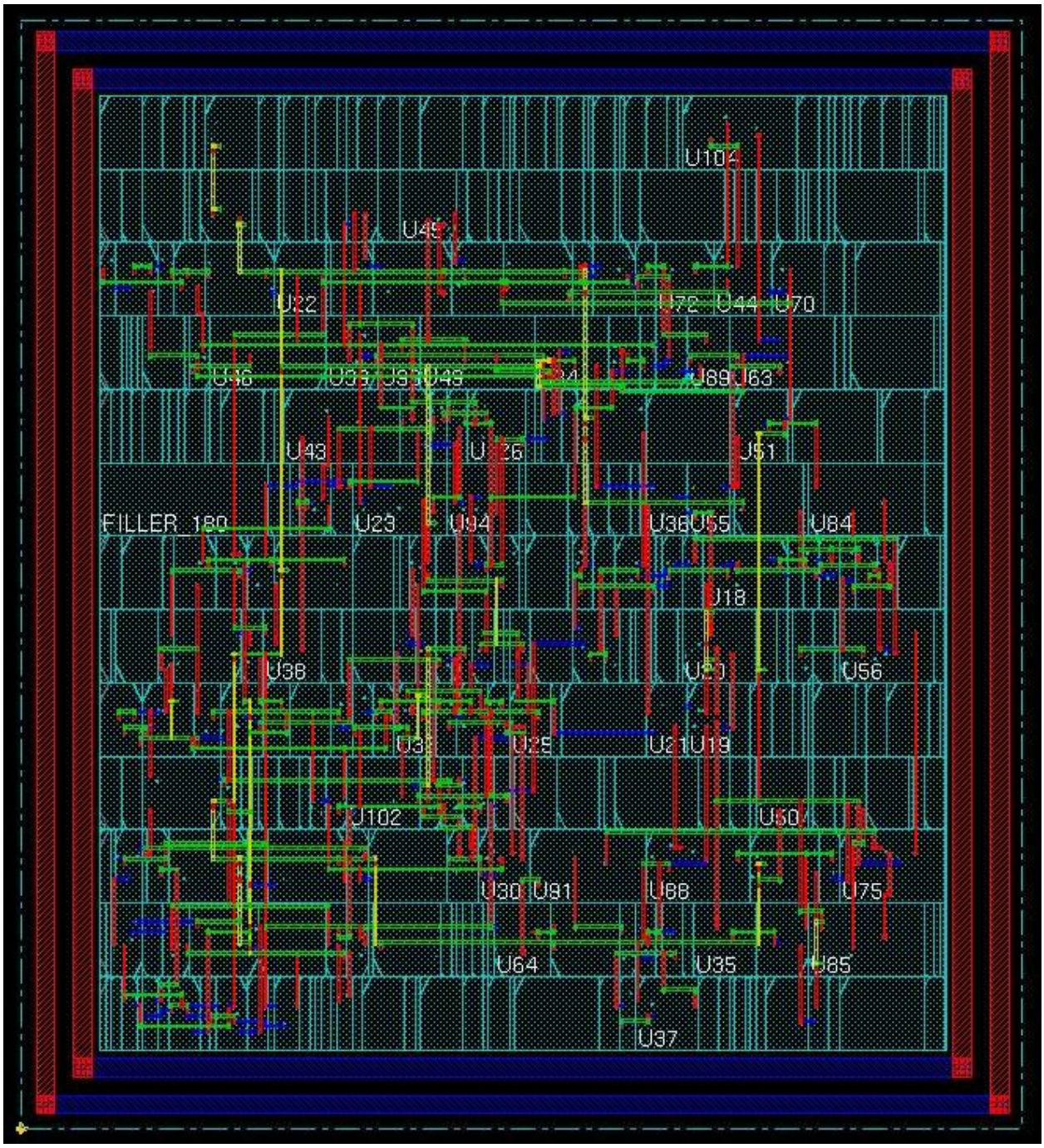


Figure 70: Layout of NCO

Table 18: 90nm NCO design information

Block name	Flip Flop Area	Combinational Area	Max Freq	Average Power
NCO	242	1079	250Mhz	2mW

4.4 Measurement Results

The results below are obtained from measurements from a 90nm implementation of CMOS 60 GHz radio. The results are presented in the form of eye diagrams and frequency spectra for different modulation schemes and input data rates.

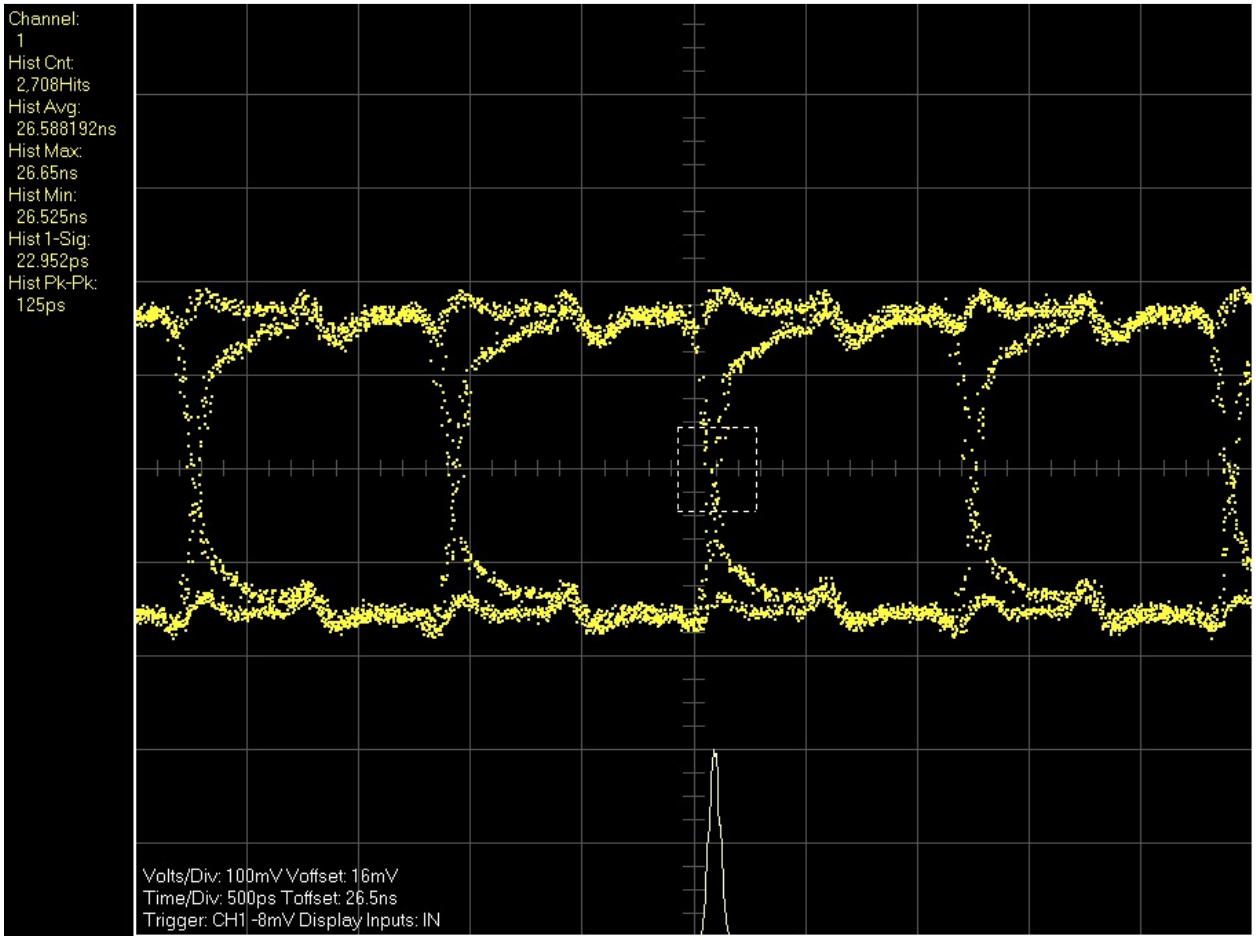


Figure 71: DBPSK Eye Diagram 864Mbps

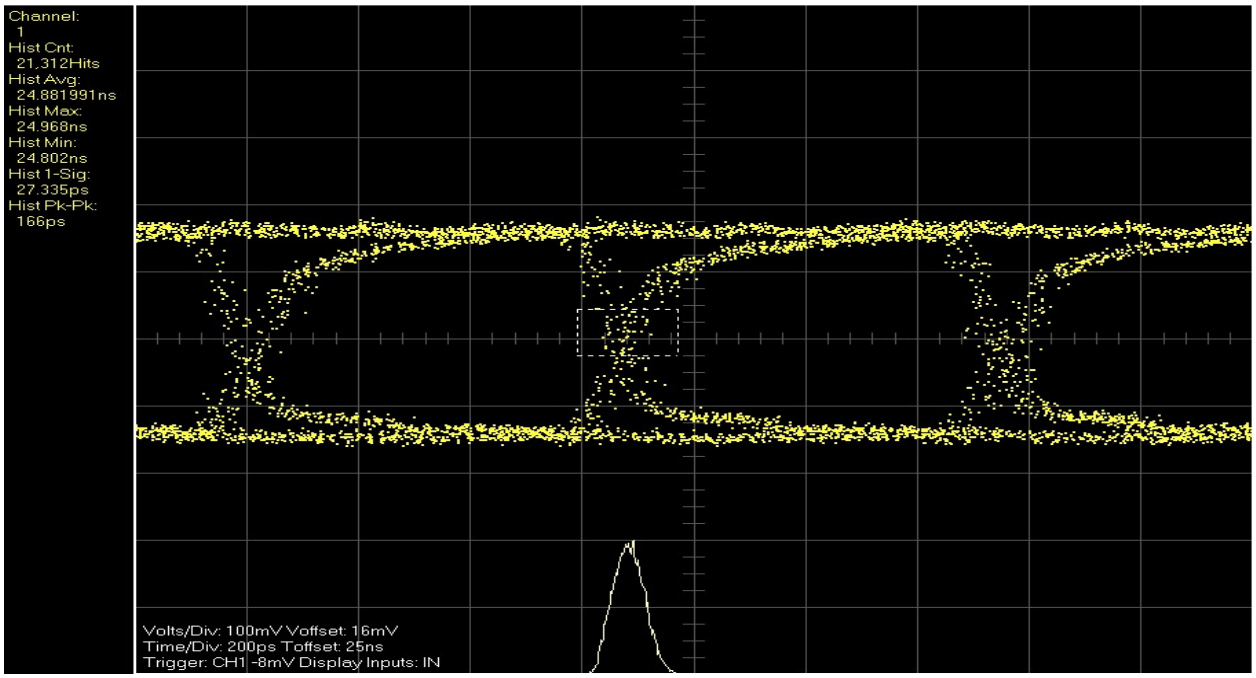


Figure 72: DBPSK Eye Diagram 1.485Gbps

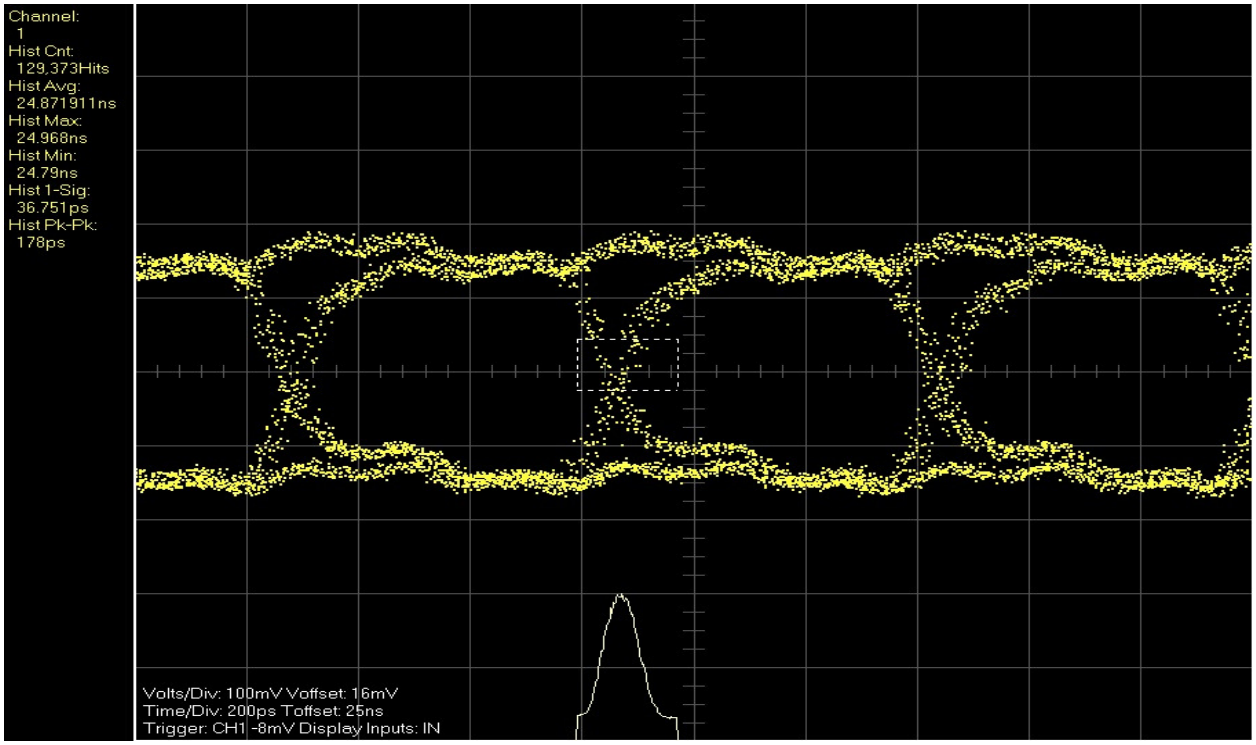


Figure 73: DBPSK Eye Diagram 1.728Gbps

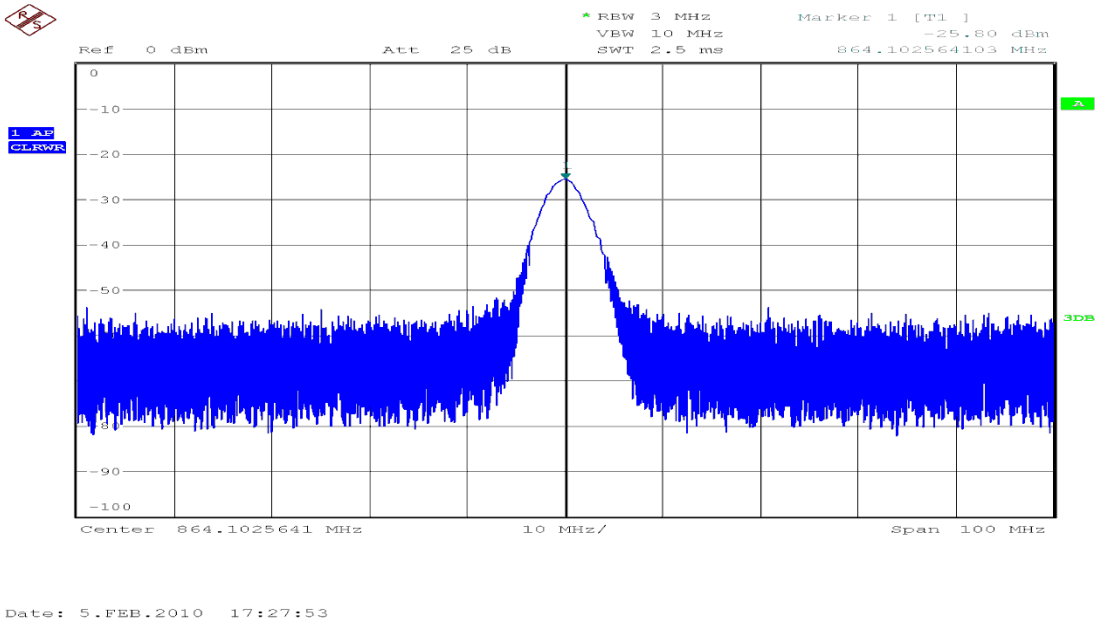


Figure 74: DBPSK Frequency Spectrum 864 MHz

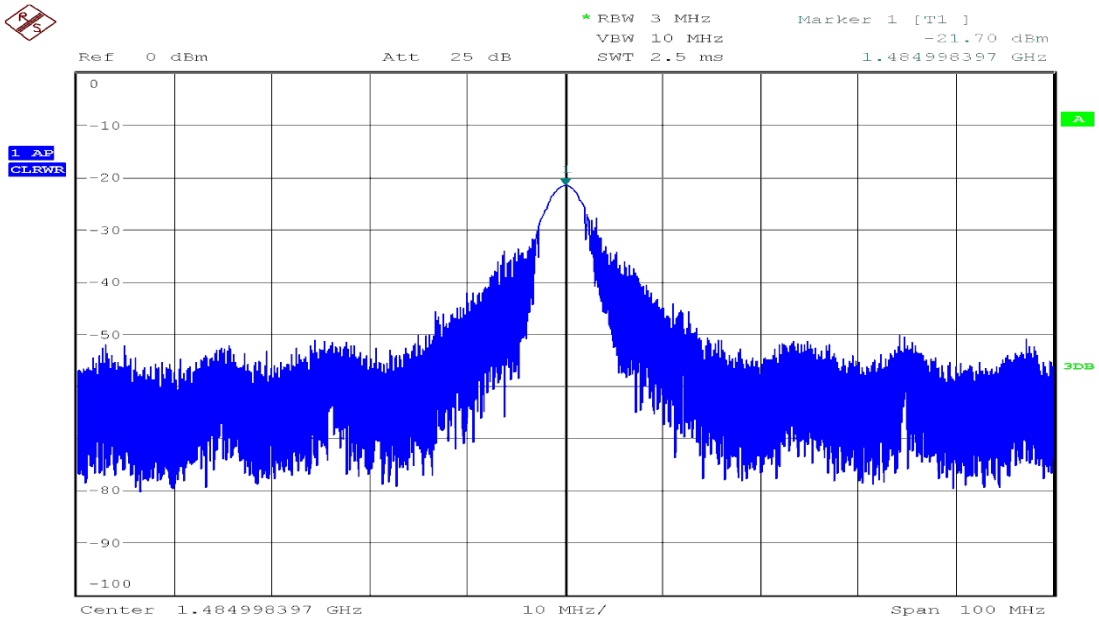


Figure 75: DBPSK Frequency Spectrum 1.484 GHz

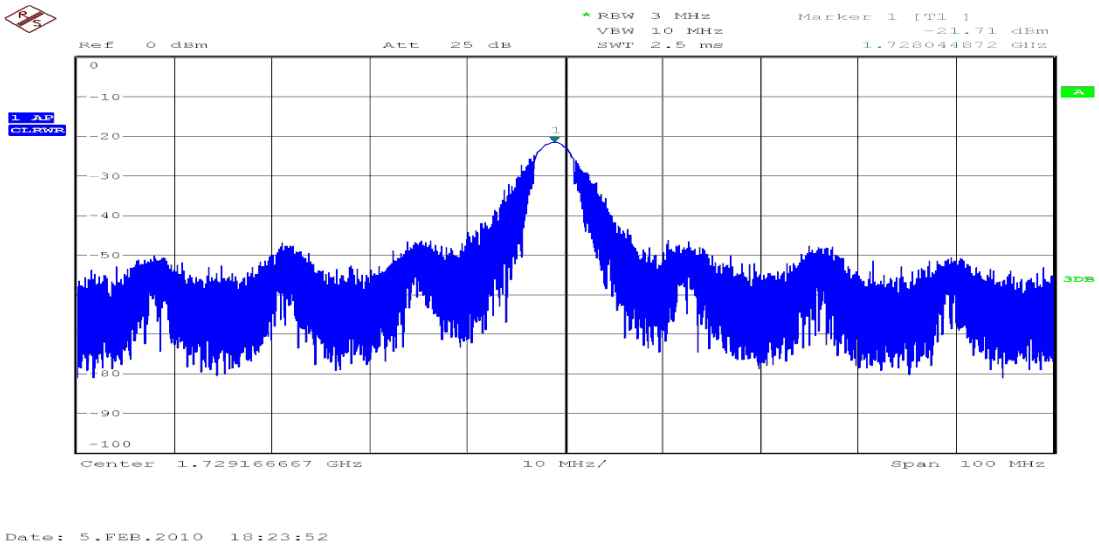


Figure 76: DBPSK Frequency Spectrum 1.728 GHz

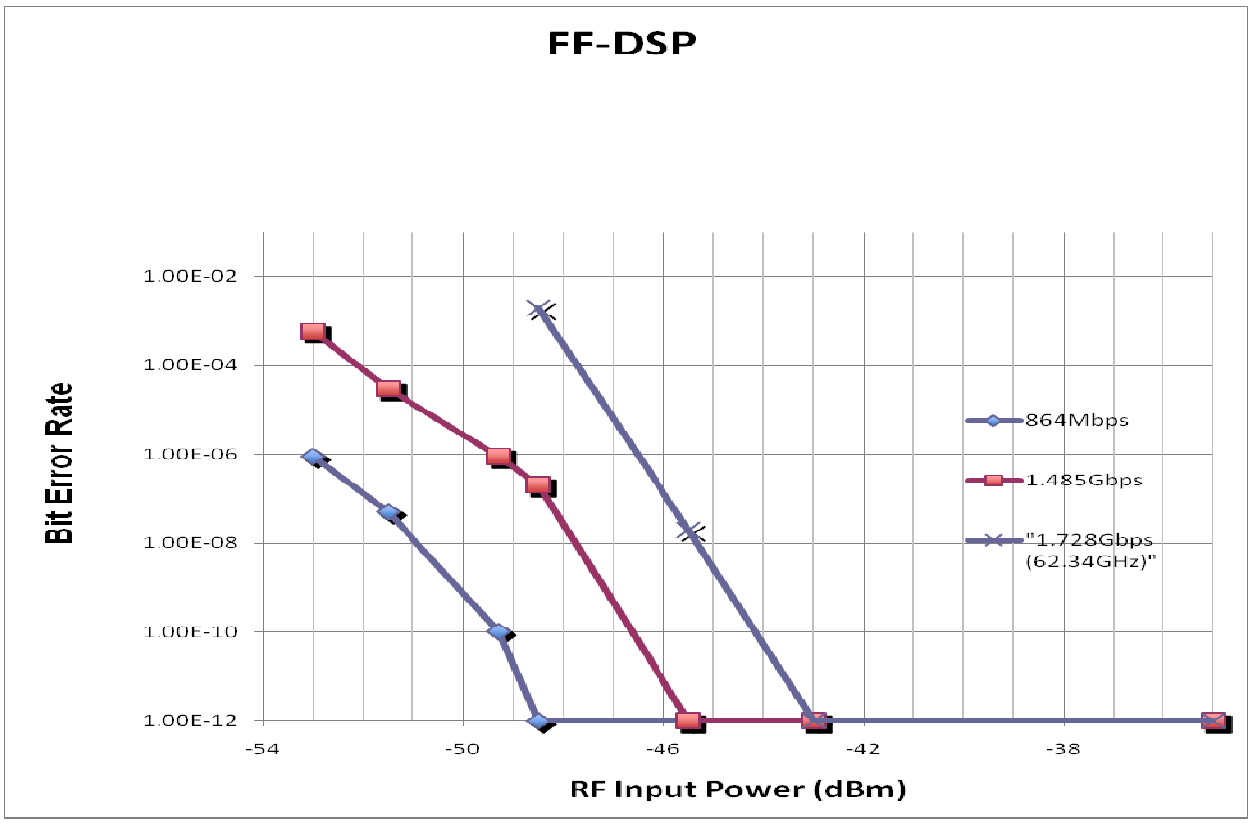


Figure 77: DBPSK BER graph

5 SERDES

SerDes [31] is a pair for functional blocks i.e. Serializer and DeSerializer, typically used in high speed communication interfaces in order to overcome the input/output limitation. These blocks convert data from Serial to parallel and vice versa. SerDes comprises of two functional blocks i.e. Parallel in Serial out or Parallel to Serial (P2S) and Serial in Parallel out or Serial to Parallel (S2P).

5.1 SerDes Architectures

SerDes architectures [31] can be grouped into 4 types:

5.1.1 Parallel Clock SerDes

Parallel Clock SerDes [31] is used to serialize parallel input along with clock and control signals. The serialized data stream is sent along with control and clock signals on separate lines. Hence it is important to balance the skew between the data line and the clock signal line in order to ensure accurate reception of the serialized data. Some of the applications of Parallel Clock SerDes include stackable Ethernet switch expansion, rack to rack and shelf-to-shelf datacom/telecom interconnects and video/camera links.

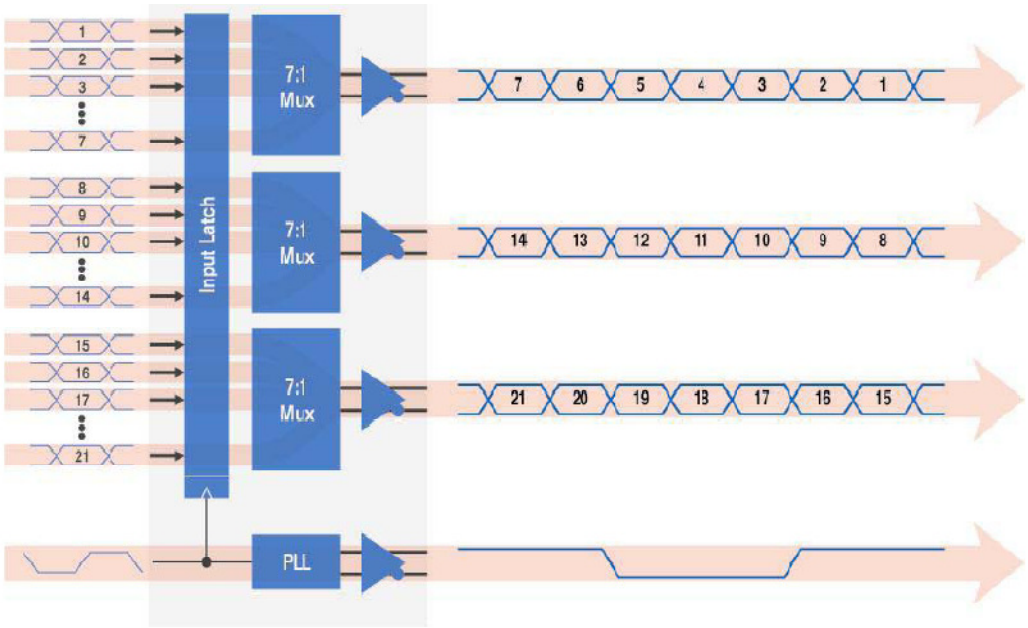


Figure 78: Parallel Clock SerDes

5.1.2 Embedded Clock SerDes

In Embedded Clock SerDes [31] data and clock signals are serialized into a single stream. Two clock bits high and low (single clock transition) are embedded into the data stream every cycle, therefore framing the start and end of each data stream. The receiver is able to synchronize with the transmitter by looking for this unique transition in the stream because although the data changes this transition remains the same. Once the receiver has identified this transition it can recover the data from the stream.

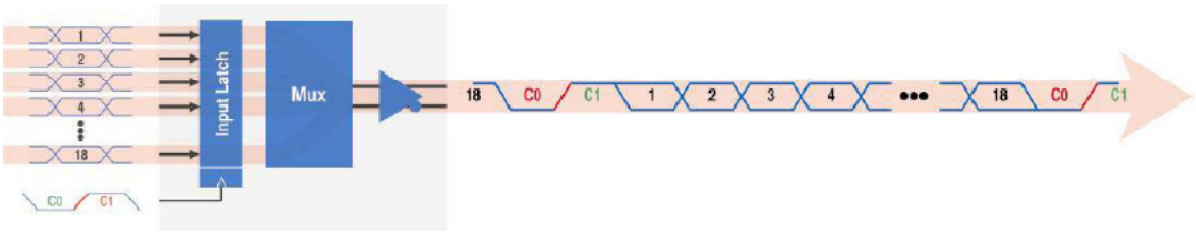


Figure 79: Embedded Clock SerDes

Embedded Clock SerDes is suited for applications that transmit raw data plus other signals such Control, parity, sync, status etc.

5.1.3 8b/10b SerDes

8b/10b SerDes [31] maps each byte of the data into 10 bits and then serializes into a serial stream. This 8bit to 10bit translation is used to ensure multiple both edge transitions in the stream to ensure DC balance. The Transmitter places boundaries in the bit stream by using a special symbol known as comma character. This unique comma bit sequence never appears in the data stream and therefore acts as a reliable data marker. The receiver looks for this marker and once alignment is achieved the receiver maps the 10bit data stream back to 8bit stream.

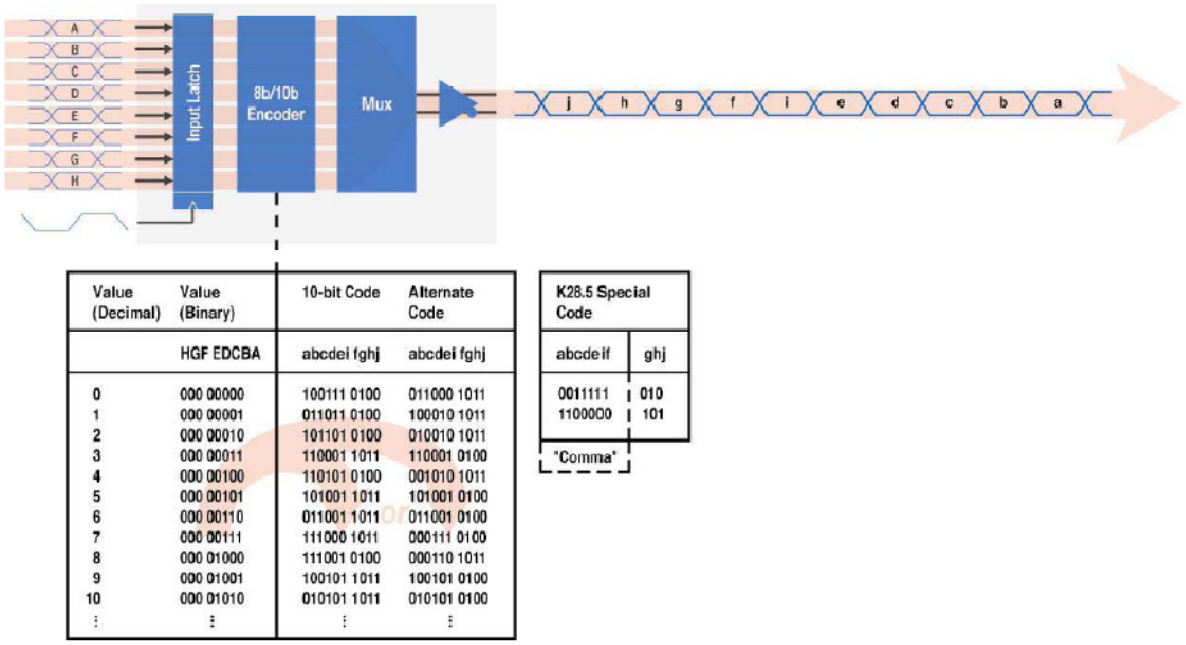


Figure 80: 8b/10b SerDes

8b/10b SerDes is well suited for byte oriented data (cell or packet) applications such as Ethernet, Fiber Channel, InfiniBand etc.

5.1.4 Bit Interleaved SerDes

Bit Interleaved SerDes [31] multiplexes several slower serial streams into faster streams at the transmitter end and demultiplexes the fast stream into its constituent slower serial streams at the receiver end. Due to the high speed nature of Bit Interleaved SerDes, it requires very precise external clocks.

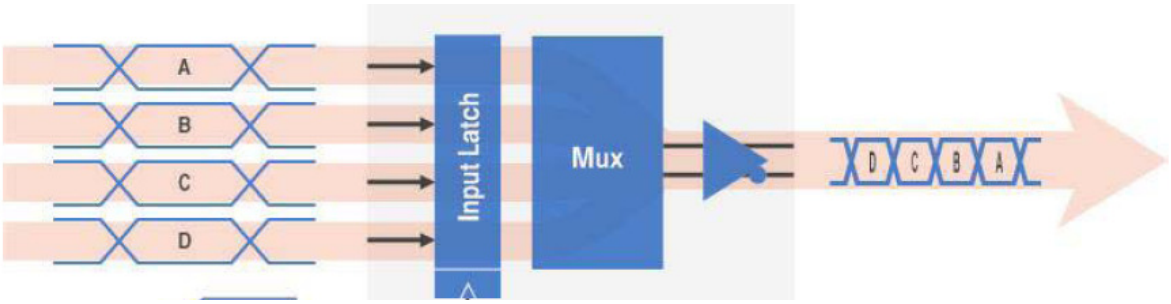


Figure 81: Bit Interleaved SerDes

Typical applications for Bit Interleaved SerDes are telecom transmission equipment such as add drop multiplexers and pseudo optical switches.

The SerDes forms an integral part of the multichip RF CMOS Radio. The SerDes chip is the interface between the RF analog front end and the Baseband chip as indicated below.

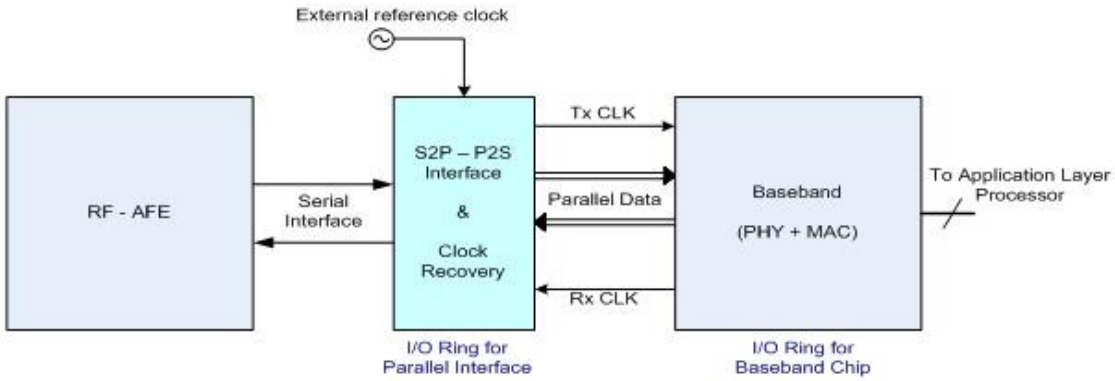


Figure 82: Multichip RF CMOS Radio

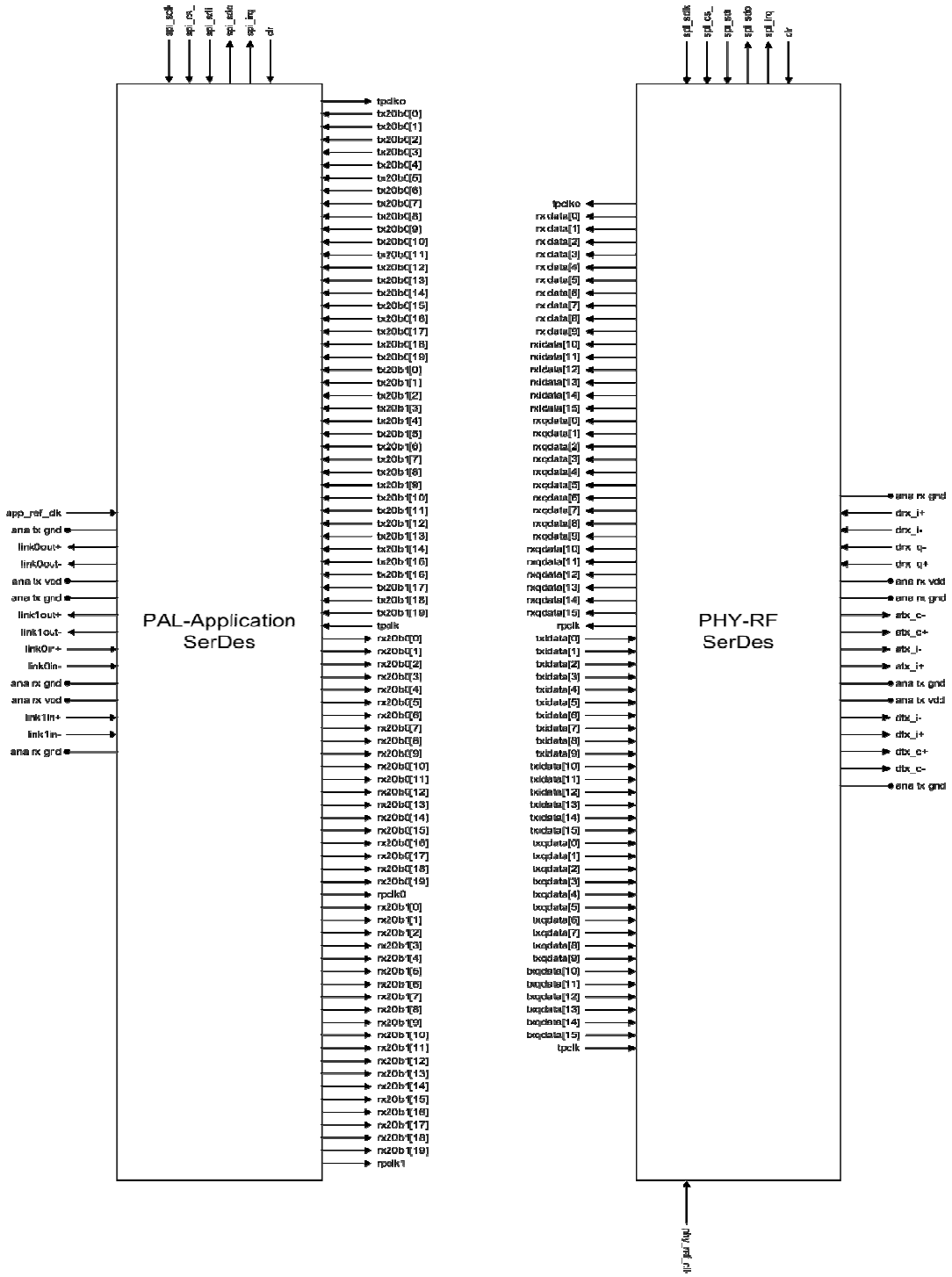


Figure 83: Block Diagram of PAL and RF SerDes interface

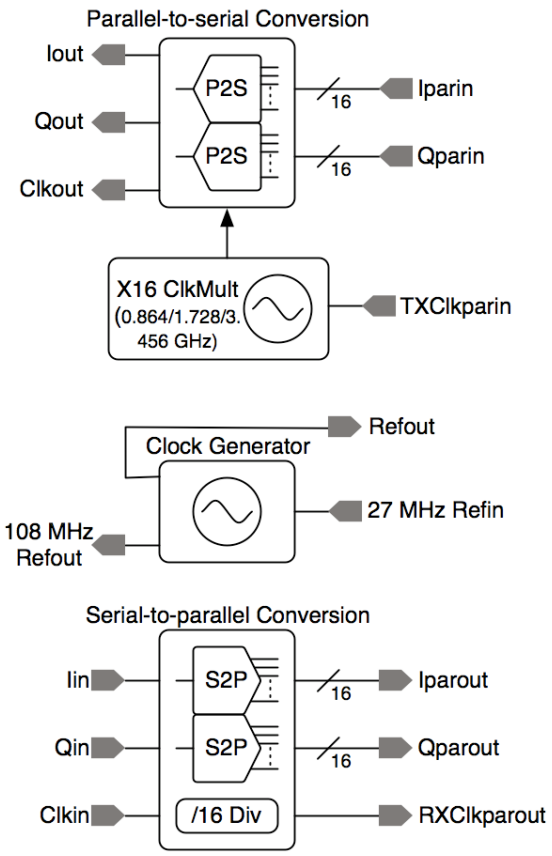


Figure 84: SerDes block diagram

The implementation details of the SerDes component modules are as follows:

5.2 1:20 Serial-to-Parallel Block

5.2.1 Description of the 20 bit S2P module

The 20 bit Serial-to-parallel module is used to convert the serial input data to parallel input data. The serial input data is synchronized with a serial clock of 2160 MHz. The output parallel data is synchronized with a parallel clock of 108 MHz.

The parallel clock is generated within the Serial-to-Parallel using a shift register based configuration. There is a 20 bit shift register used in counter mode i.e. a 1 is shifted through the shift register. The shift register is clocked by a serial clock. Divide by 20 is achieved when the 1 is shifted through all the flops of the shift register.

The data is also shifted through a shift register. The shift register is clocked using the Serial clock i.e. 2160 MHz. The 20 bit parallel data is output depending on the load signal generated using the parallel clock generated by the shift register based clock divider logic.

5.2.2 Functional Diagram

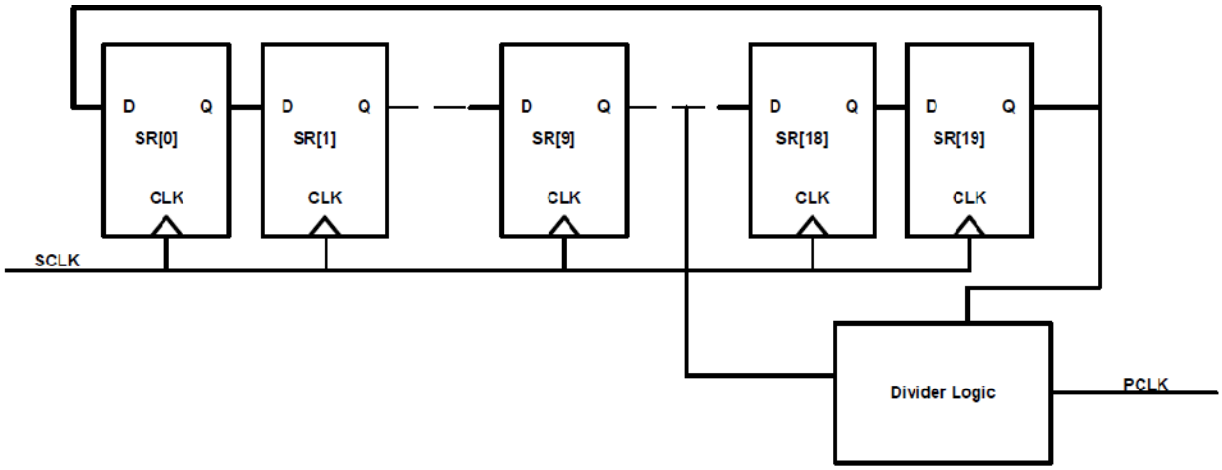


Figure 85: Shift Register Based Clock Divider Logic

5.2.3 Timing Diagram

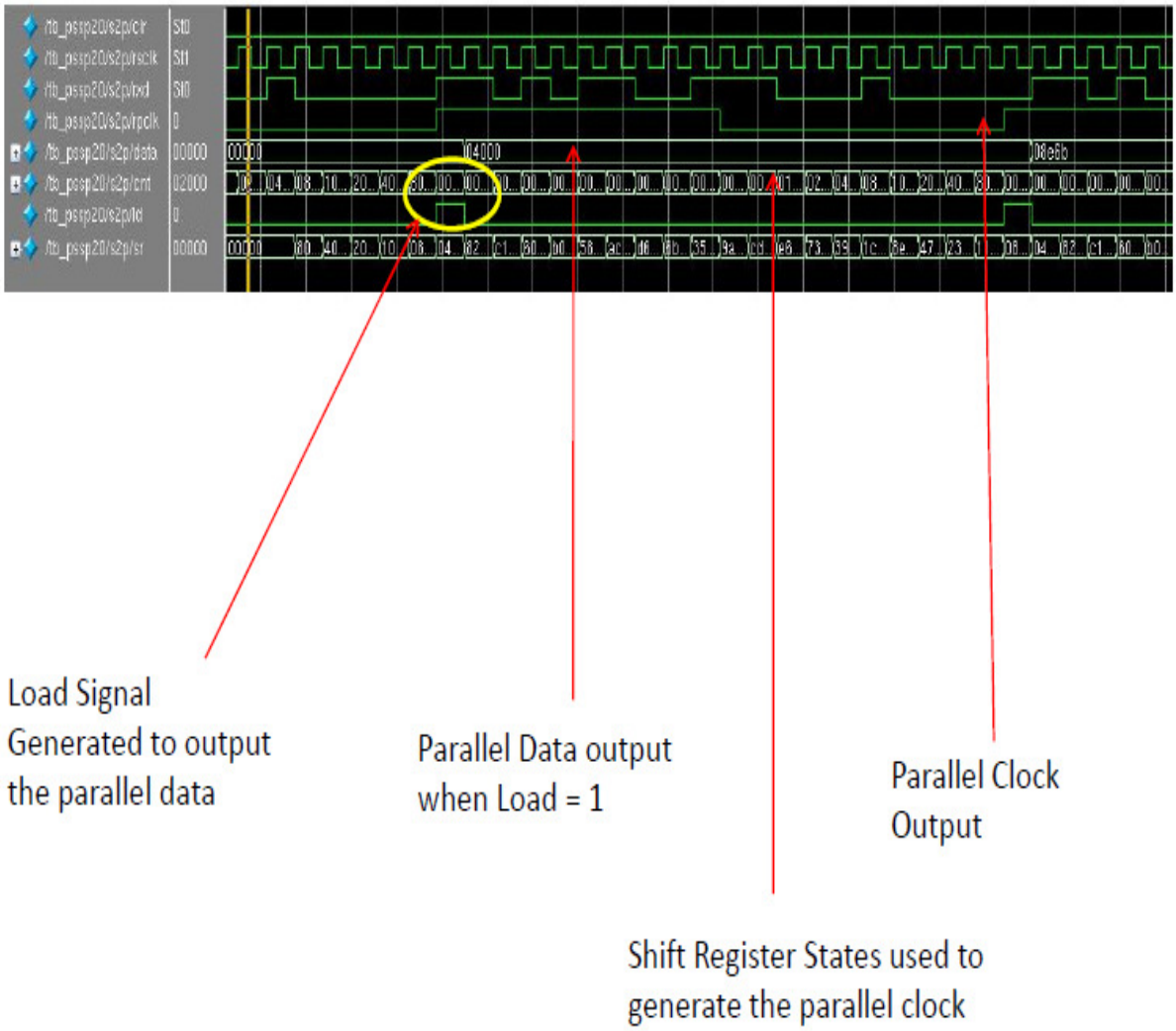


Figure 86: Timing diagram for 20 bit S2P

5.2.4 Layout

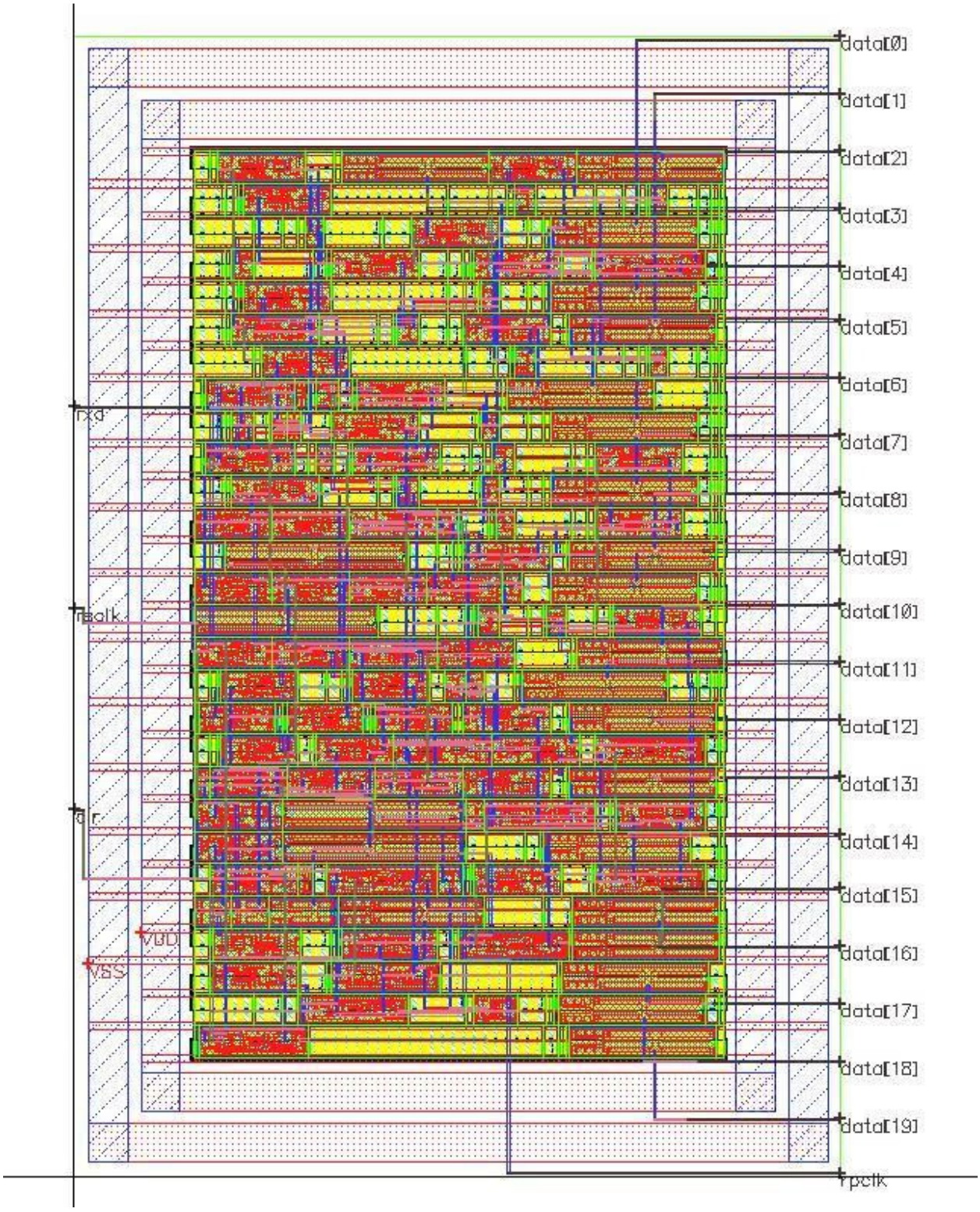


Figure 87: 20 bit S2P Layout

5.3 20:1 Parallel-to-Serial Block

5.3.1 Description of the 20 bit P2S module

The 20 bit P2S block, reads in 20 bits of parallel data, and shifts the data out 1 bit at a time clocked by the serial clock i.e. 2160 MHz The P2S block requires both the parallel clock i.e.108MHz and the Serial clock i.e. 2160 MHz.

In order to avoid any misalignment between the edges of the parallel clock and serial clock, a synchronizer structure is formed by using two back to back flip flops clocked by serial clock. The input to the synchronizer is the parallel clock. The output of the synchronizer is the parallel clock synchronized with the serial clock. This synchronized parallel clock is used to generate the load signal. When the load signal is high (1'b1) the parallel data from the input is loaded onto the shift register. When the load signal is low (1'b0), the lowest bit of the shift register is shifted out to the serial output pin.

5.3.2 Functional Diagram

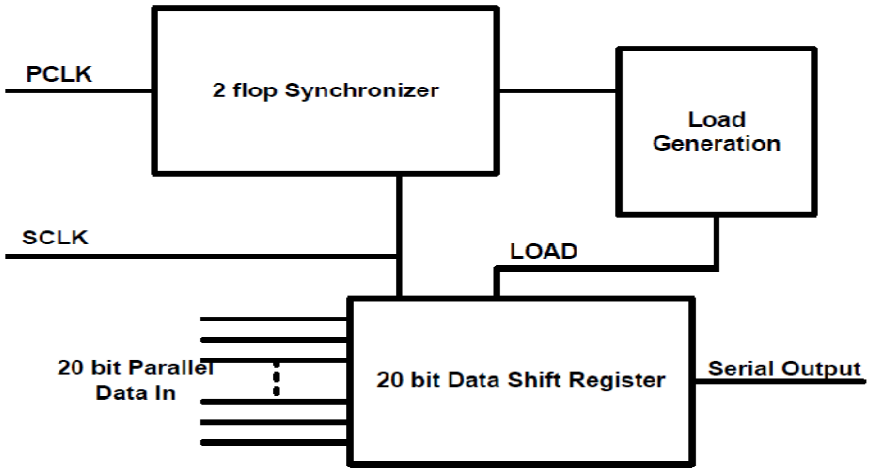


Figure 88: Block Diagram 20 bit Parallel-to-Serial

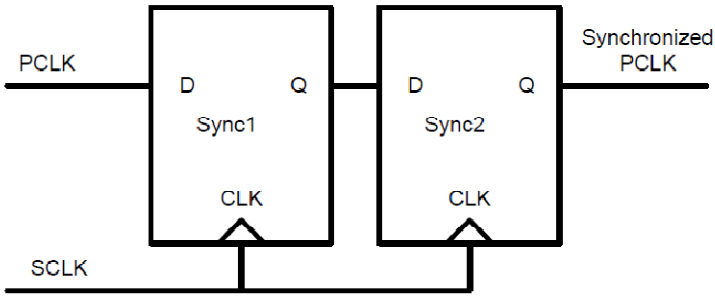


Figure 89: Synchronizer between parallel and serial clock

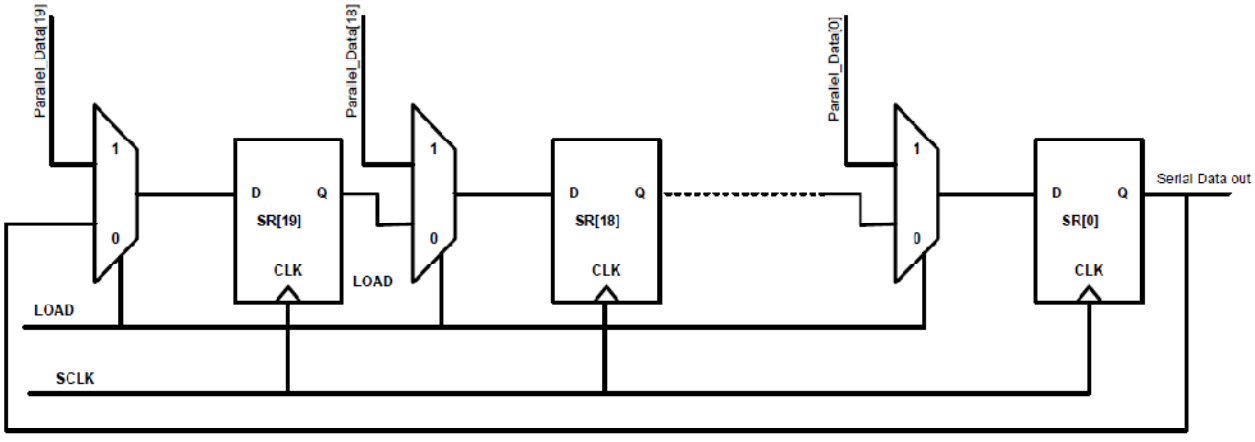


Figure 90: 20 bit Data Shift Register with Parallel Load

5.3.3 Timing Diagram

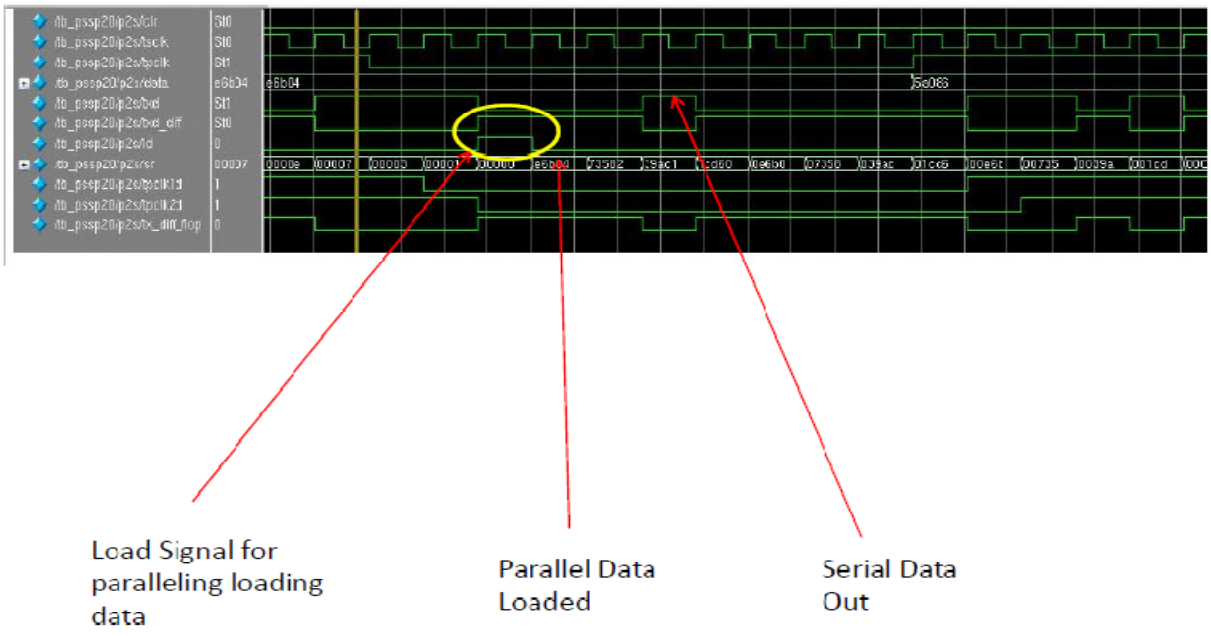


Figure 91: Timing Diagram for 20 bit P2S

5.3.4 Layout

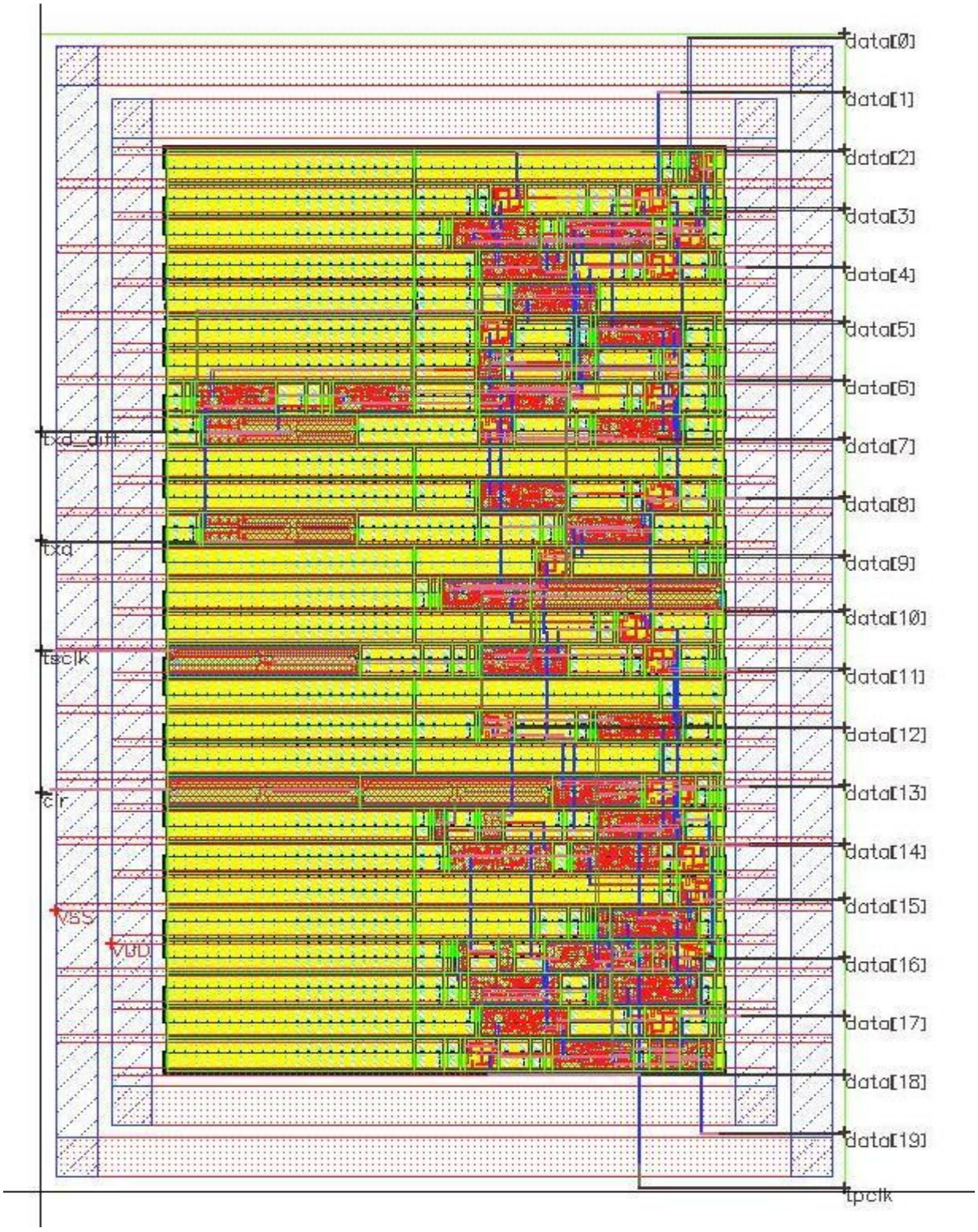


Figure 92: 1:16 Serial-to-Parallel Block

5.3.5 1:16 Serial-to-Parallel Block

5.4 Description of the 16 bit S2P module

The 16 bit Serial-to-parallel module is used to convert the serial input data to parallel input data. The serial input data is synchronized with a serial clock of 1728 MHz. The output parallel data is synchronized with a parallel clock of 108 MHz.

The parallel clock is generated within the Serial-to-Parallel using a shift register based configuration. There is a 16 bit shift register used in counter mode i.e. a 1 is shifted through the shift register. The shift register is clocked by the serial clock. Divide by 16 is achieved when the 1 is shifted through all the flops of the shift register.

The data is also shifted through a shift register. The shift register is clocked using the Serial clock i.e. 1728 MHz. The 16 bit parallel data is output depending on the load signal generated using the parallel clock generated by the shift register based clock divider logic.

5.4.1.1 Functional Diagram

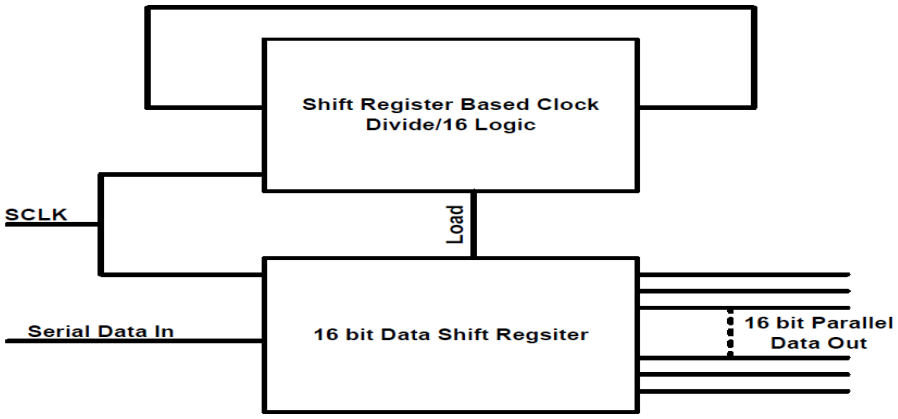


Figure 93: Block Diagram 16 bit Serial-to-Parallel

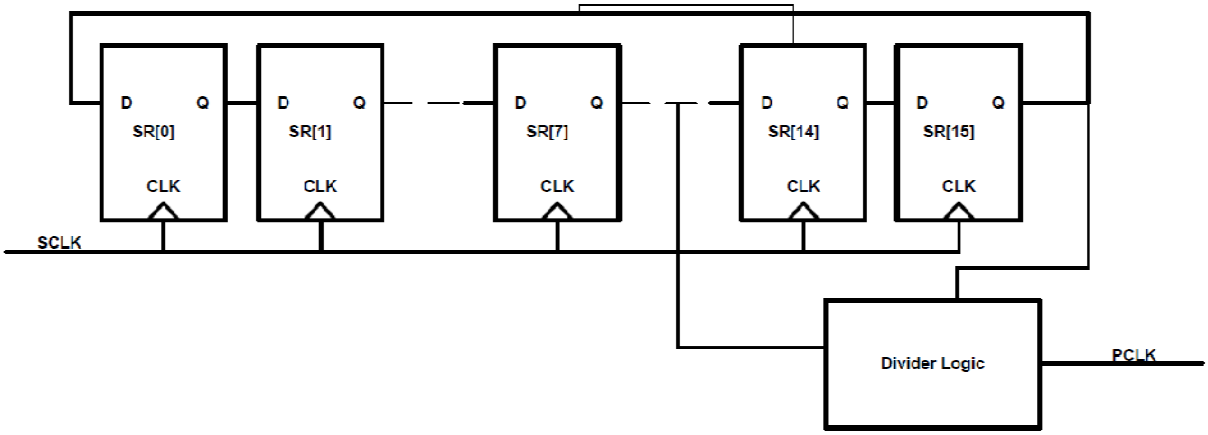


Figure 94: Shift Register Based Clock Divider Logic

5.4.2 Timing Diagram

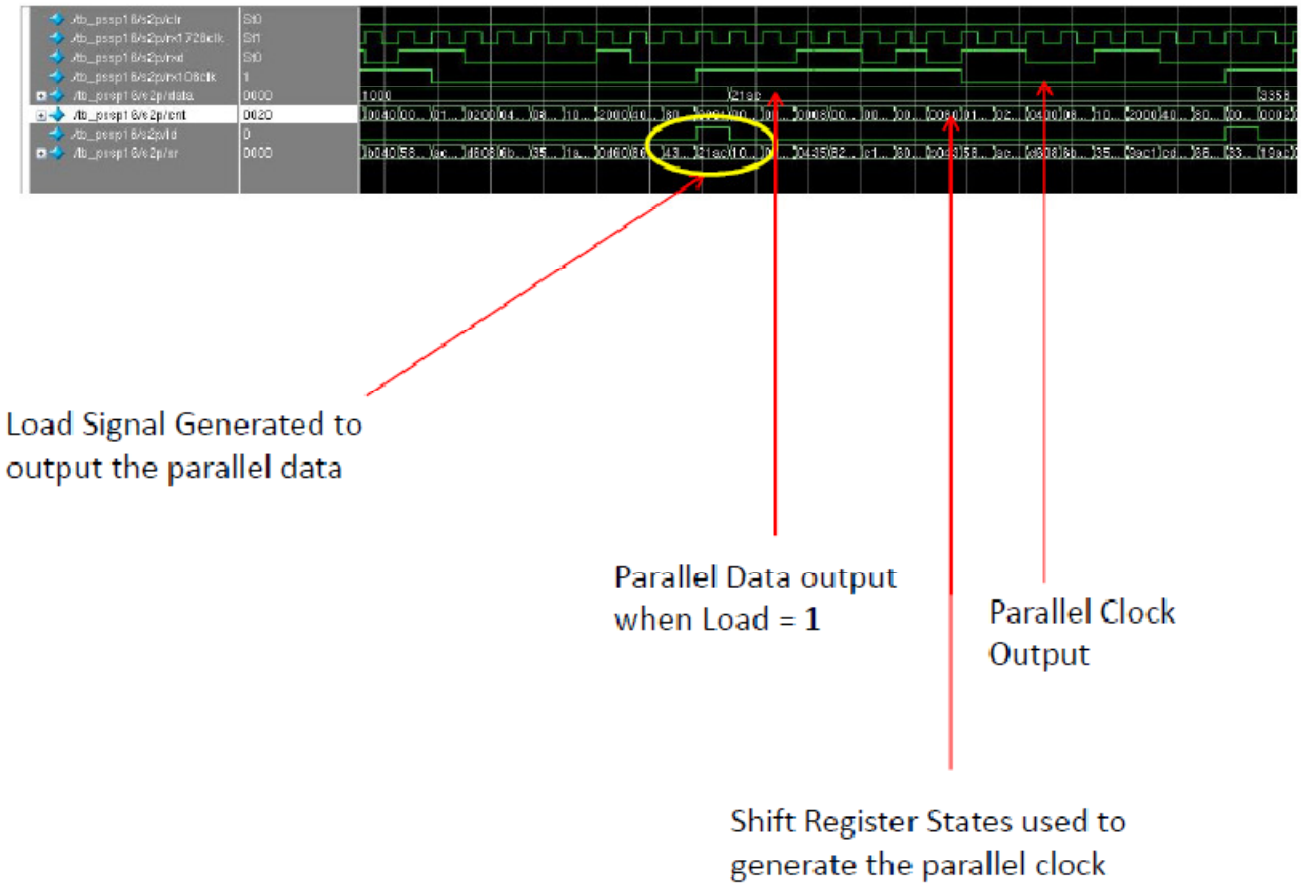


Figure 95: Timing Diagram for 16 bit S2P

5.4.3 Layout

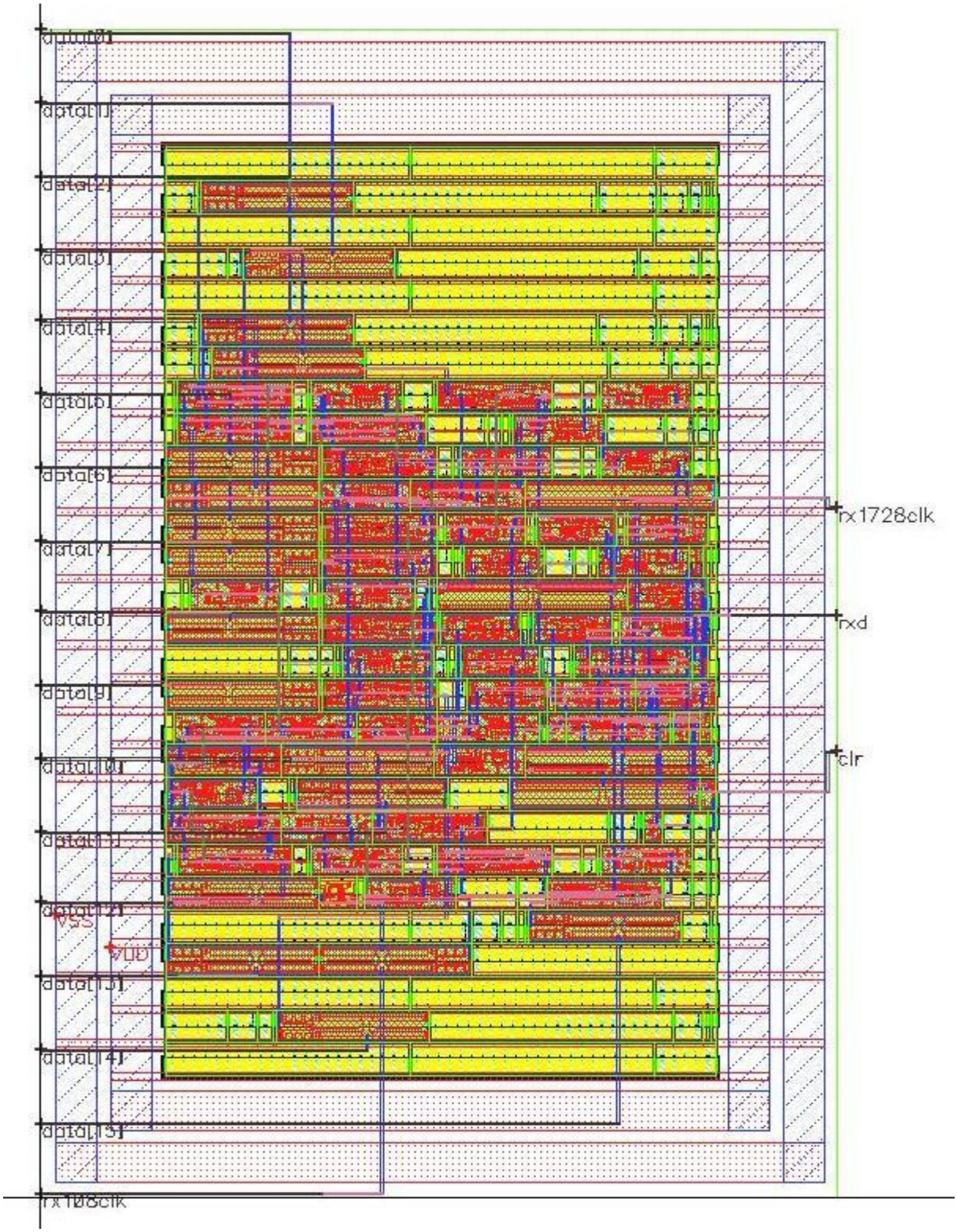


Figure 96: Layout of 16 bit S2P

5.5 16:1 Parallel-to-Serial Block

5.5.1 Description of the 16 bit P2S module

The 16 bit P2S block, reads in 16 bits of parallel data, and shifts the data out 1 bit at a time clocked by the serial clock i.e. 1728 MHz. The P2S block requires both the parallel clock 108MHz and the Serial clock 1728 MHz.

In order to avoid any misalignment between the edges of parallel clock and serial clock, a synchronizer structure is formed by using two back to back flip flops clocked by serial clock. The input to the synchronizer is the parallel clock. The output of the synchronizer is the parallel clock synchronized with the serial clock. This synchronized parallel clock is used to generate the load signal. When the load signal is high (1'b1) the parallel data from the input is loaded onto the shift register. When the load signal is low (1'b0), the lowest bit of the shift register is shifted out to the serial output pin.

5.5.2 Functional Diagram

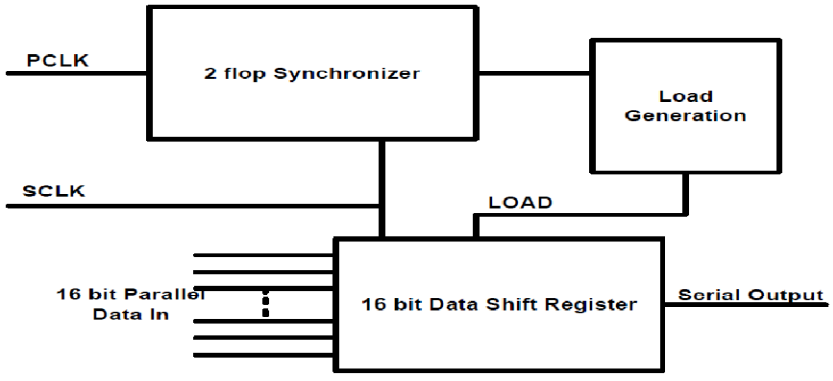


Figure 97: Block Diagram 16 bit Parallel-to-Serial

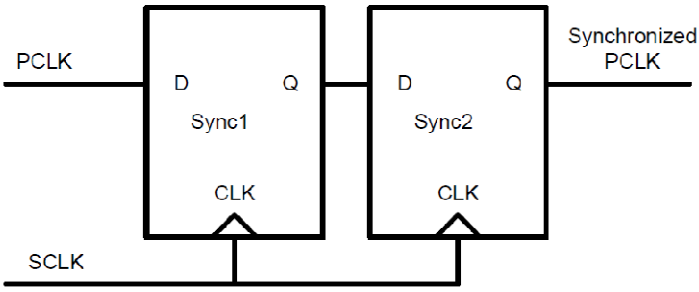


Figure 98: Synchronizer between parallel and serial clock

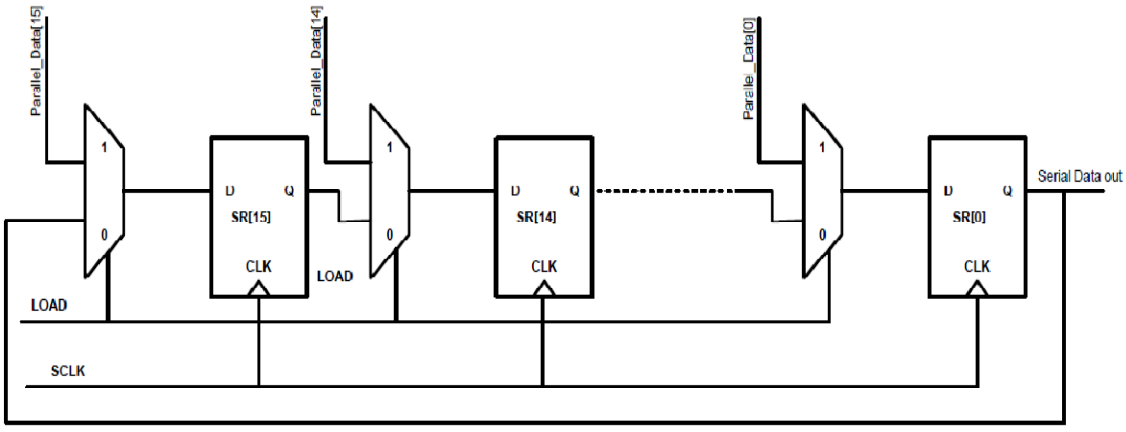


Figure 99: 16 bit Data Shift Register with Parallel Load

5.5.3 Timing Diagram

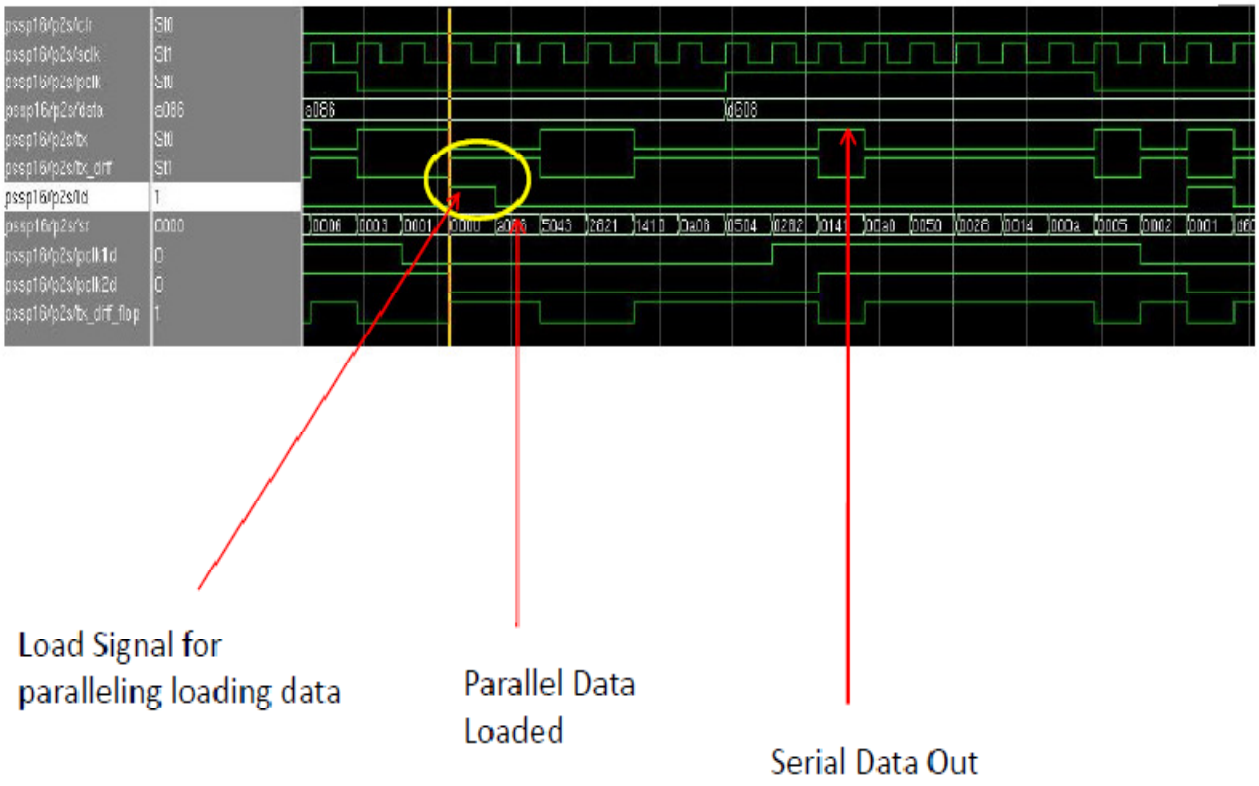


Figure 100: Timing Diagram for 16 bit P2S

5.5.4 Layout

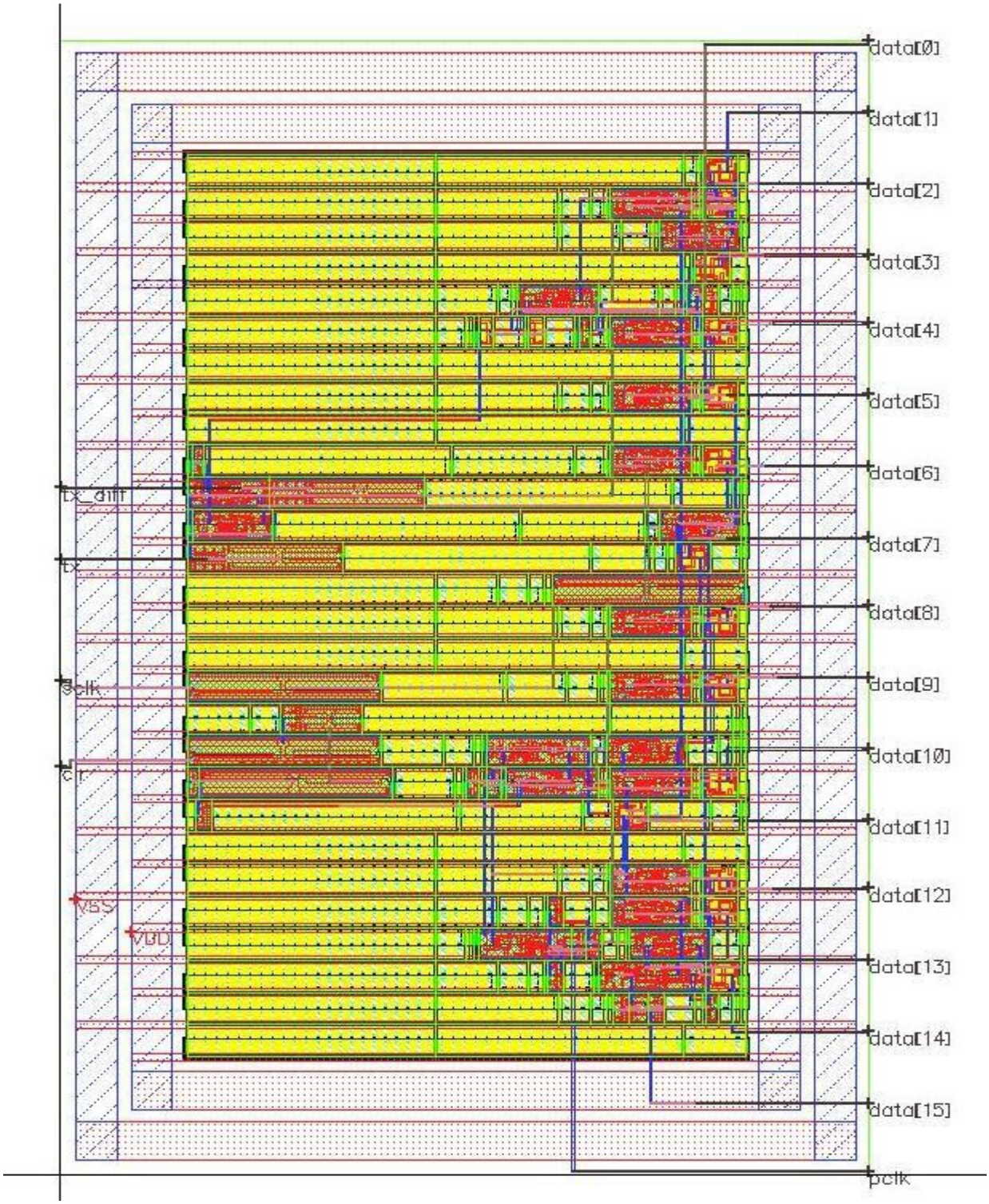


Figure 101: Layout of 16 bit P2S

5.6 SERDES Layout.

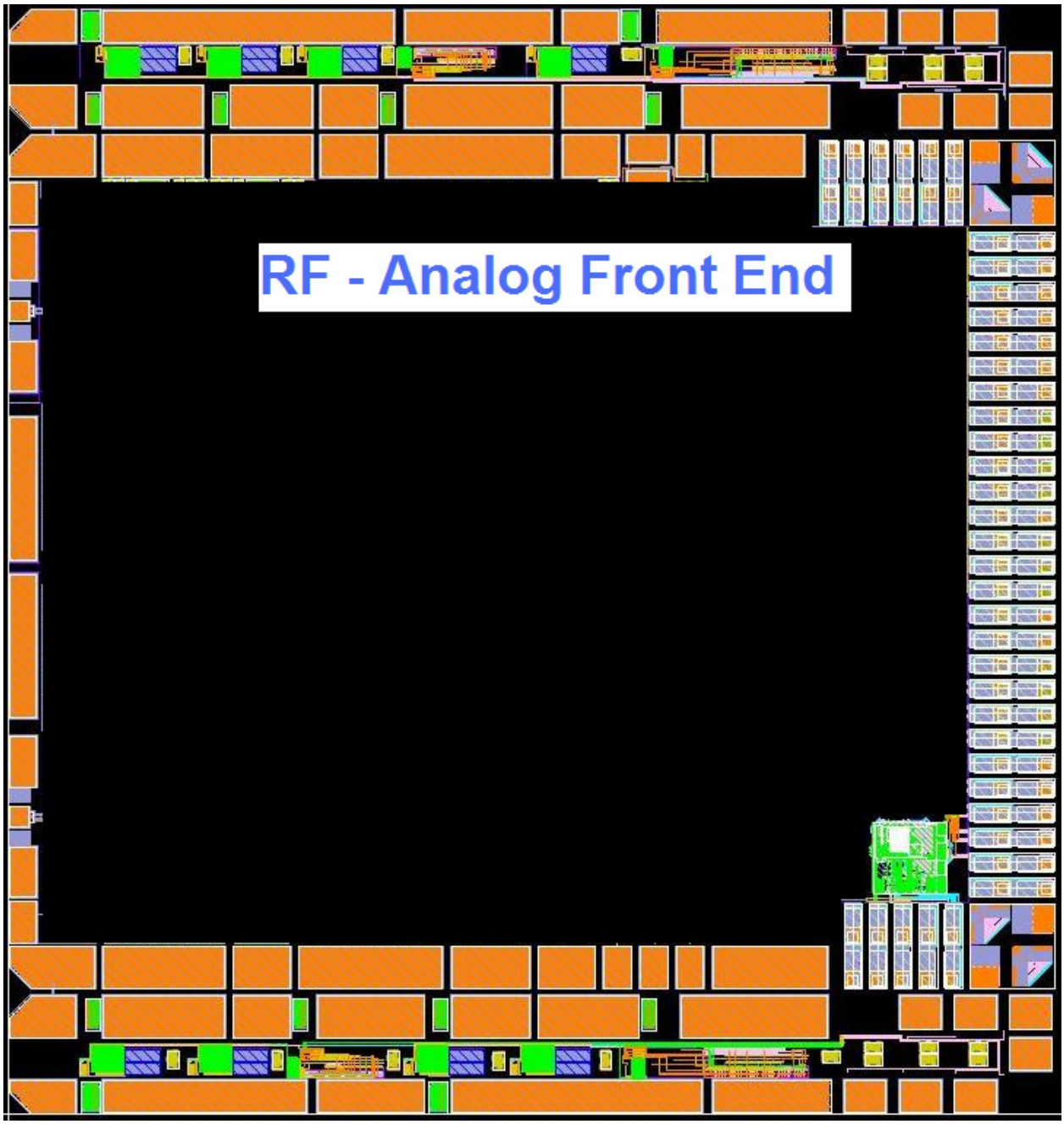


Figure 102: RF Analog Front End IO diagram with SerDes Interface on the right

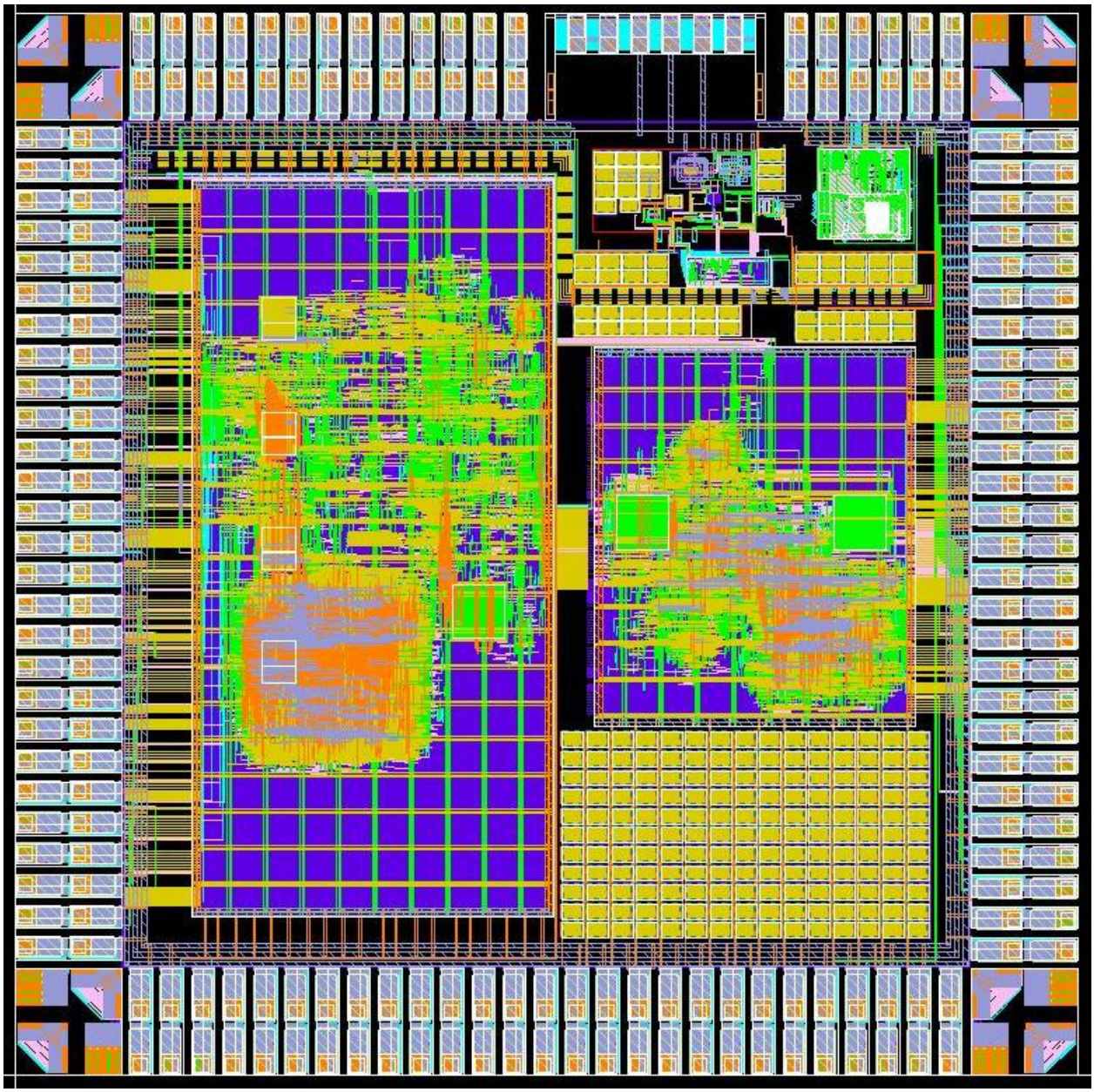


Figure 103: Baseband Layout with SerDes IO Interface

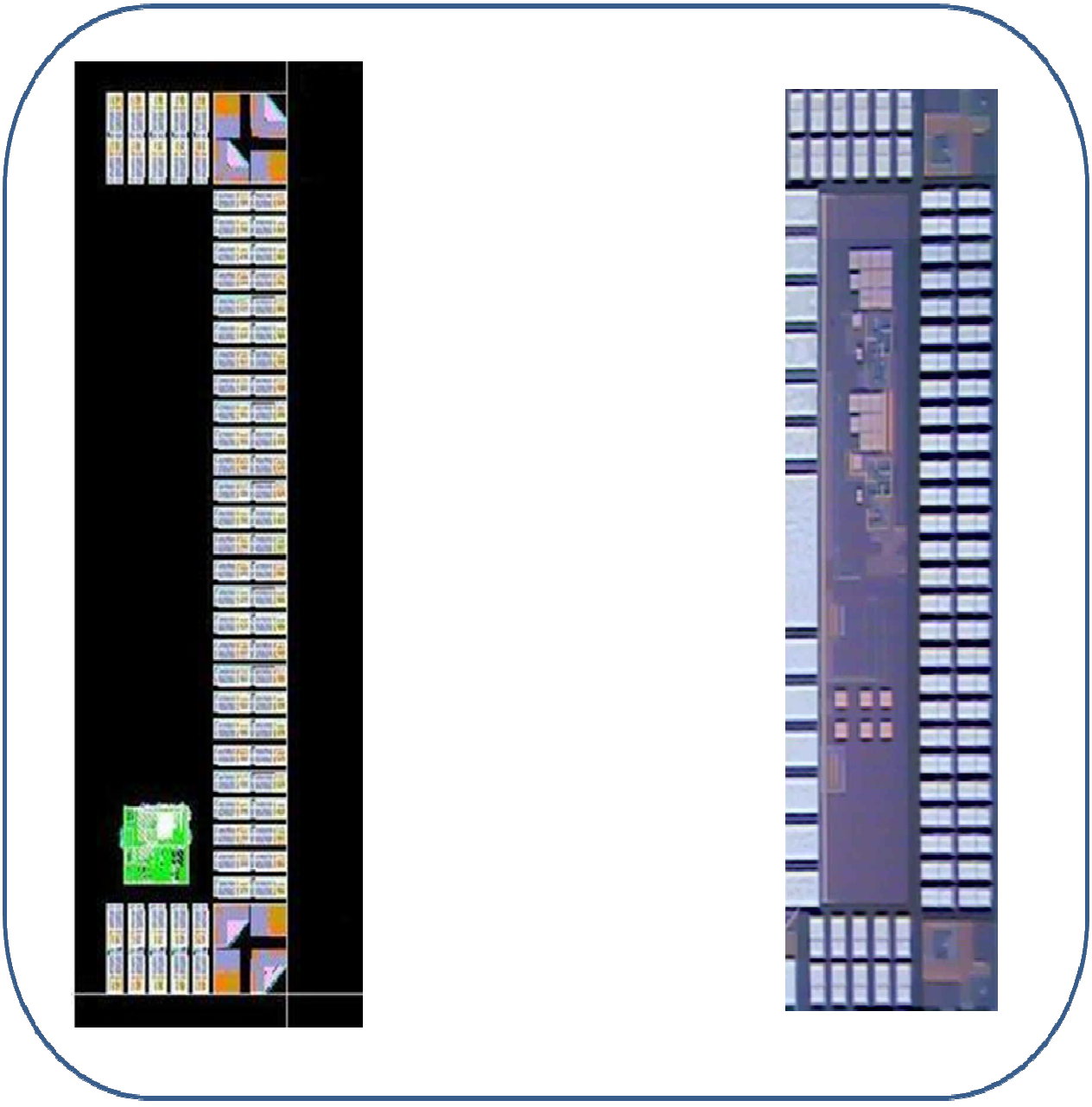


Figure 104: SERDES IO diagram (left) Complete SerDes layout with IO Interface (right)

6 CONCLUSION

In this work an ASIC design methodology is presented. The ASIC methodology includes all aspects of ASIC design i.e. from design specification to Layout verification. The ASIC methodology presented is used to implement the PHY and MAC modules using 90nm process technology as part of the Baseband chip. The PHY and MAC modules have a combined gate count of over 1 million gates. A basic standard cell library for both 90nm and 45nm technology node is also presented. The use of this standard cell library in the custom design of more complex logic is also presented.

In the latter part of the work a coherent and non coherent demodulation technique is presented. The demodulator is part of the RF chip. The RF chip is implemented using a 90nm process technology. The measurement results from the demodulator are also presented in the form of eye diagrams and frequency spectra for different input data rates. It can be seen from the measurement results that the demodulator works at multi gigabit data rates. Since the RF input is a serial data stream and the Baseband chip operates on parallel data, a SerDes interface is required between baseband and RF chip. The SerDes interface converts the parallel data from baseband to serial stream to be used in the RF chip and converts the serial data from the RF chip to parallel data to be used in the baseband chip in a synchronous manner. Design and Implementation details of two versions of the SerDes are also presented i.e. 16bit and 20bit.

REFERENCES

[1] Ali M. Niknejad, Hossein Hashemi, “mm-Wave silicon technology: 60GHz and beyond”

[2] C. E. Shannon , “The Mathematical Theory of Communication”

[3] R. V. L. Hartley ,"Transmission of Information"

[4] 60GHz:Achieving the ultimate wireless dream : http://www.wirelessnetdesignline.com/howto/207400754 , (Date 05/04/2010)

[5] Niknejad, A.M.; Emami, S.; Heydari, B.; Bohsali, M.; Adabi, E, “Nanoscale CMOS for mm- Wave Applications” in Compound Semiconductor Integrated Circuit Symposium, 2007. CSIC 2007. IEEE

[6] http://wireless.fcc.gov/outreach/2004broadbandforum/comments/YDI_benefits60GHz.pdf (Date 05/04/2010)

[7] http://www.st.com/stonline/products/technologies/asic/method.htm (Date 05/04/2010)

[8] 1364-2005 IEEE Standard for Verilog Hardware Description Language: http://ieeexplore.ieee.org/servlet/opac?punumber=10779 (Date 05/04/2010)

[9] IEEE Standard VHDL Language Reference Manual : http://ieeexplore.ieee.org/servlet/opac?punumber=4772738 (Date 05/04/2010)

[10] Modelsim : http://model.com/content/modelsim-pe-simulation-and-debug (Date 05/04/2010)

[11] Ncsim : http://www.cadence.com/products/ld/design_team_simulator/pages/default.aspx (Date 05/04/2010)

[12] Design Compiler user guide : https://solvnet.synopsys.com/dow_retrieve/D-2010.03/dcug/ni/dcug.pdf (Date 05/04/2010)

[13] Liberty format : : ftp://ftp.synopsys.com/pub/Liberty_User_Guides_Reference_Manual_2009.06/Liberty_User_Guides_Reference_Manual_2009.06.pdf (Date 05/04/2010)

[14] SOC Encounter User guide : http://support.cadence.com/wps/PA_DocumentViewer/pubs/soceUG/soceUG8.1.3/soceUG.pdf (Date 05/04/2010)

[15] Synopsys Design Constraints (SDC) http://www.synopsys.com/Community/Interoperability/Pages/TapinSDC.aspx (Date 05/04/2010)

[16] LEF/DEF 5.5 Language Reference : http://support.cadence.com/wps/PA_DocumentViewer/pubs//lefdefref/lefdefref5.5/lefdefref.pdf (Date 05/04/2010)

[17] SPEF standard : IEEE Standard for Integrated Circuit (IC) Delay and Power Calculation System –Description http://standards.ieee.org/reading/ieee/std/dasc/1481-1999.pdf (Date 05/04/2010)

[18] SDF : Standard Delay Format Specification : http://www.eda.org/sdf/sdf_3.0.pdf (Date 05/04/2010)

[19] Prime Time User Guide : https://solvnet.synopsys.com/dow_retrieve/D-2010.03/ptsi/ni/ptsi.pdf (Date 05/04/2010)

[20] Power Compiler User Guide : https://solvnet.synopsys.com/dow_retrieve/D-2010.03/pwcug/ni/pwcug.pdf (Date 05/04/2010)

[21] SAIF(Switching Activity Interchange format) : http://www.synopsys.com/community/interoperability/pages/tapinsaif.aspx (Date 05/04/2010)

[22] DFM (Design for Manufacturability : http://en.wikipedia.org/wiki/Design_for_manufacturability_(IC) (Date 05/04/2010)

[23] LVS (Layout vs. Schematic) : http://en.wikipedia.org/wiki/Layout_Versus_Schematic (Date 05/04/2010)

[24] GDSII stream format : http://www.buchanan1.net/stream_description.shtml (Date 05/04/2010)

[25] LabVIEW based advanced instrumentation systems By S. Sumathi, P. Surekha

[26] "Practical Costas loop design by Jeff Feigin : http://rfdesign.com/images/archive/0102Feigin20.pdf (Date 05/04/2010)

[27] Mathew P. Donoadio, “CIC filter Introduction” : http://www.mikrocontroller.net/attachment/51932/cic2.pdf (Date 05/04/2010)

[28] Chappell, M. , McEwan, A, “A low power high speed accumulator for DDFS applications” in Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS '04

[29] Comb Filter : http://en.wikipedia.org/wiki/Comb_filter (Date 05/04/2010)

[30] First-Order Digital Filters for Second-Order Digital PLLs : http://home.netcom.com/~chip.f/plls.htm (Date 05/04/2010)

[31] Dave Lewis, SerDes Architectures and Applications : http://www.national.com/appinfo/lvds/files/designcon2004_serdes.pdf (Date 05/04/2010)

[32] DRC : http://en.wikipedia.org/wiki/Design_rule_checking (Date 05/04/2010)

[33] Khosrow Golshan, “ Physical Design Essentials: An ASIC Design Implementation Perspective”

[34] Raymond Charles, Vincent Macario, “Modern personal radio systems”

[35] Eric Hagemann , “The Costas Loop - An Introduction” : http://archive.chipcenter.com/dsp/DSP010315F1.html (Date 05/04/2010)

[36] http://www.commsdesign.com/design_corner/showArticle.jhtml?articleID=18310339 (Date 05/04/2010)

[37] CIC filter : http://en.wikipedia.org/wiki/Cascaded_integrator-comb_filter (Date 05/04/2010)